DECchip 21050 PCI-to-PCI Bridge

Data Sheet

Order Number: EC–Q9XVB–TE

Revision/Update Information: This document supersedes the *DECchip 21050 PCI-to-PCI Bridge Data Sheet*, EC–Q9XVA–TE.

Digital Equipment Corporation
Maynard, Massachusetts

December 1994

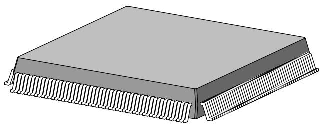
This document was prepared using VAX DOCUMENT Version 2.1.

DECchip 21050 PCI-to-PCI Bridge

Data Sheet

21050 Features:

- Implements revision 2.0 PCI-compliant drivers
- Supports two 32-bit PCI buses
- Provides maximum clock frequency of 33 MHz
- Provides concurrent primary and secondary bus operation
- Forwards memory read and write transactions in either direction
- Forwards I/O read and write transactions in either direction
- Forwards configuration read and write transactions in downstream direction
- Converts configuration write transactions to special cycles in either direction



- Supports memory transaction filtering through two programmable memory address regions — one prefetchable and one non-prefetchable
- Provides I/O transaction filtering through one programmable I/O address region
- Provides seven secondary PCI clock outputs
- Provides programmable rotating arbitration function which supports up to six secondary bus masters

The DECchip 21050 PCI-to-PCI Bridge is a low-cost, high-performance chip that expands the electrical capacity of all PCI systems (for example, Alpha AXP, Pentium, x86). The 21050 allows motherboard designers to add more PCI devices or more PCI option card slots than a single PCI bus can support. Option card designers can use the 21050 to implement multiple device PCI option cards.

Technical Support and Information

North America:	**1-800-332-2717**
TTY:	**1-800-332-2515**
Outside North America:	**+1-508-568-6868**

Chip Ordering Information

North America:	**1-800-344-4825 (1-800-DIGITAL)**
Outside North America:	**Contact your local Digital sales office**

Contents

5 Configuration Register Description

6 Diagnostics and Test Mechanisms

7 DECchip 21050 Specifications

A Technical Support, Ordering, and Associated Literature

Figures

Tables

Preface

Purpose

This data sheet describes the DECchip 21050 PCI-to-PCI bridge chip (21050). The 21050 expands the electrical capacity of all PCI systems (for example, Alpha AXP, Pentium, x86, etc.). The 21050 allows motherboard designers to add more PCI devices or more PCI option card slots than a single PCI bus can support. Option card designers can use the 21050 to implement multiple device PCI option cards.

This document is not intended to describe the details of PCI protocol. For more information, see the associated literature section at the back of this document.

Audience

This document is for chip designers who need to expand the electrical capacity of their PCI bus architectures.

Manual Organization

This manual consists of seven chapters and an appendix.

- Chapter 1 is an overview of the 21050.
- Chapter 2 lists the 21050 pin assignment in alphabetic and numeric order.
- Chapter 3 provides a description of pin signals.
- Chapter 4 describes chip functions, including bus interfaces, transaction types, transaction termination, address decoding, arbitration, and other functions.
- Chapter 5 provides programmer reference information about configuration space registers.

- Chapter 6 provides diagnostics information.
- Chapter 7 lists the electrical characteristics of the chip.
- Appendix A contains information about DECchip technical support, ordering, and associated literature.

Conventions

The following conventions are used in this document.

Convention	Meaning
Note	Provides general information.
Caution	Provides information to prevent damage to equipment.
Warning	Provides information to prevent personal injury.
***x*trdy_l**	PCI signals on the initiator (master) bus are denoted by *x* preceding the PCI signal name.
***y*trdy_l**	PCI signals on the target (slave) bus are denoted by *y* preceding the PCI signal name.

1
Introduction

The DECchip 21050 PCI-to-PCI Bridge (also called the 21050) is a low-cost, high-performance chip that expands the electrical capacity of all PCI systems (for example, Alpha AXP, Pentium, x86). The 21050 allows motherboard designers to add more PCI devices or more PCI option card slots than a single PCI bus can support. Option card designers can use the 21050 to implement multiple device PCI option cards.

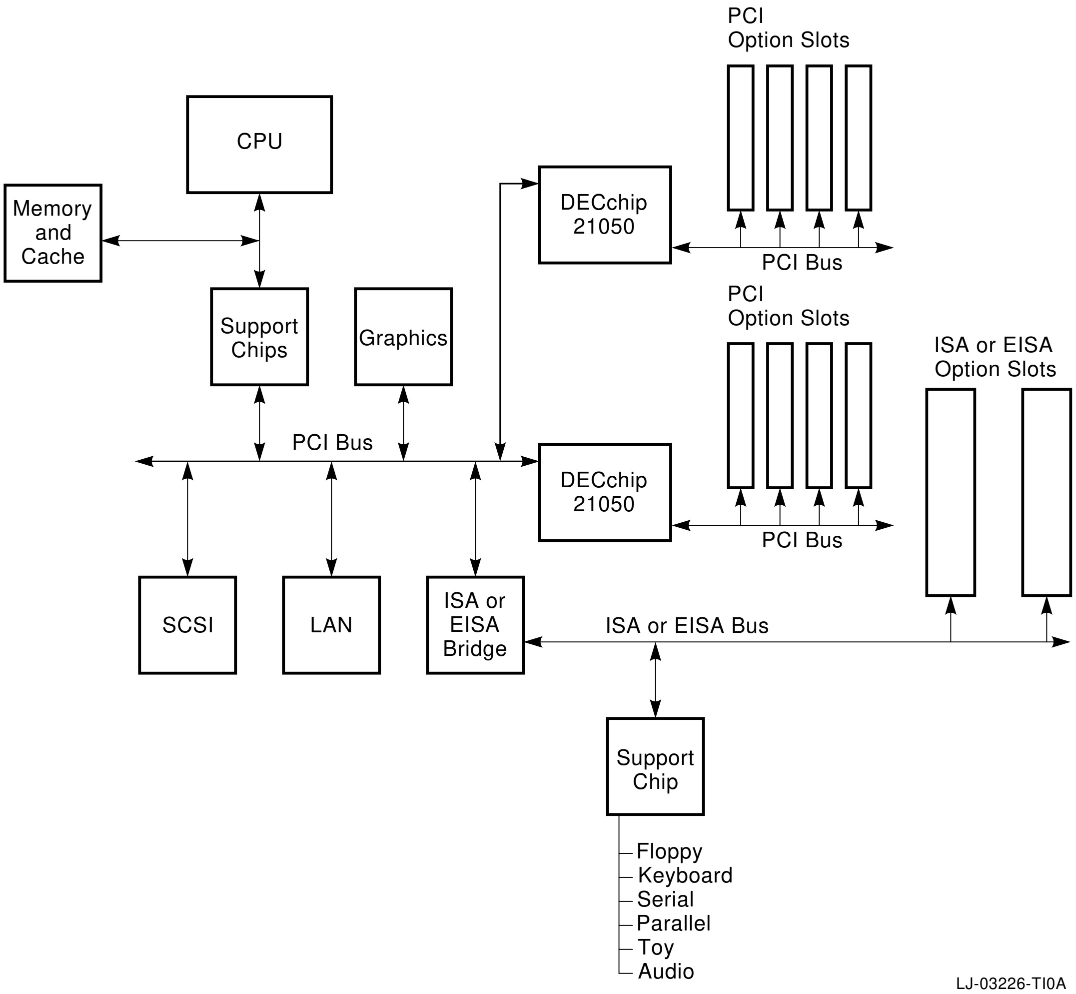
1.1 General Description

The DECchip 21050 has two PCI interfaces. The primary PCI interface connects directly to the PCI bus closest to the host CPU. The secondary PCI interface creates a new and independent PCI bus. The primary function of the bridge is to allow transactions to occur between a master on one PCI bus and a target on the other PCI bus.

The 21050 also allows the two PCI buses to operate independently. A master and a target located on the same PCI bus can communicate with each other even if the other PCI bus is busy. As a result, the 21050 can isolate traffic between devices on one PCI bus from devices on other PCI buses. This is a major benefit to system performance in some applications such as multimedia.

The 21050 can extend a system beyond the electrical loading limits of a single PCI bus. Each new PCI bus created by the addition of a 21050 provides support for additional electrical loads. Motherboard designers can use the 21050 to add more PCI devices or PCI option card connectors to the motherboard. Figure 1–1 shows the 21050 on the system board.

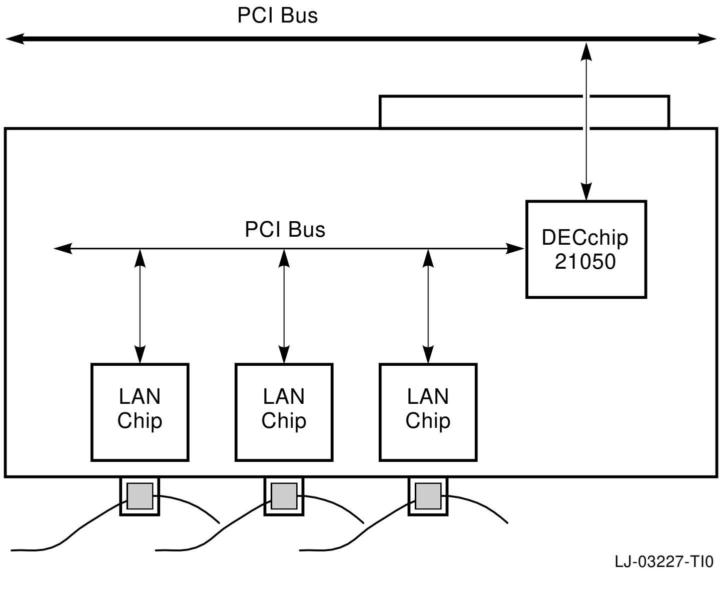
Figure 1–1 DECchip 21050 PCI-to-PCI Bridge on the System Board



Option card designers can use a 21050 to implement multiple device PCI option cards. Without a 21050, you can attach only one PCI device to the PCI option connector (the *PCI Local Bus Specification* restricts PCI option cards to a single connection per PCI signal in the option card connector). In this application, the 21050 creates an independent PCI bus on the option card to which many devices can be attached.

Figure 1–2 shows the 21050 with option cards.

Figure 1–2 DECchip 21050 PCI-to-PCI Bridge with Option Cards



1.2 Features

The 21050 has the following features:

- Implements revision 2.0 PCI-compliant drivers
- Supports two 32-bit PCI buses
- Provides maximum clock frequency of 33 megahertz
- Provides concurrent primary and secondary bus operation
- Conditionally forwards the following transactions:
  - Memory read and write transactions in either direction
  - I/O read and write transactions in either direction
  - Configuration read and write transactions in the downstream direction
  - Configuration write transactions to special cycles in either direction
- Supports memory transaction filtering through two programmable memory address regions—one prefetchable and one non-prefetchable
- Supports read prefetching for memory read transactions
- Provides up to eight dwords (32 bytes) of write posting for memory write transactions
- Provides I/O transaction filtering through one programmable memory I/O address region
- Provides ISA-mode for I/O transaction filtering
- Provides two programmable video graphics adapter (VGA) bits that support forwarding of VGA memory and I/O addresses, or forwarding of VGA palette I/O writes
- Provides master latency timers and target wait timers, for each PCI interface, which limit the amount of latency on either bus
- Provides concurrent resource lock operation
- Propagates locks across the 21050
- Provides seven secondary PCI bus clock outputs

- Enables the following central functions through **s_cfn_l** input pin for secondary bus:
  - Programmable rotating arbitration function supporting up to six secondary bus masters
  - Secondary PCI bus parking at the 21050
- Provides pins for buffer empty status and write posting control
- Supports **perr** and **serr** signals with error checking functionality

1.3 Architecture Overview

The 21050 is implemented as an ASIC in a standard 5-volt complementary metal-oxide semiconductor (CMOS) process. The 21050 is packaged in a 208-pin plastic quad flat-pack.

Figure 1–3 shows the major functions of the 21050, and Table 1–1 describes the major sub-blocks.

Figure 1–3 DECchip 21050 PCI-to-PCI Bridge Block Diagram

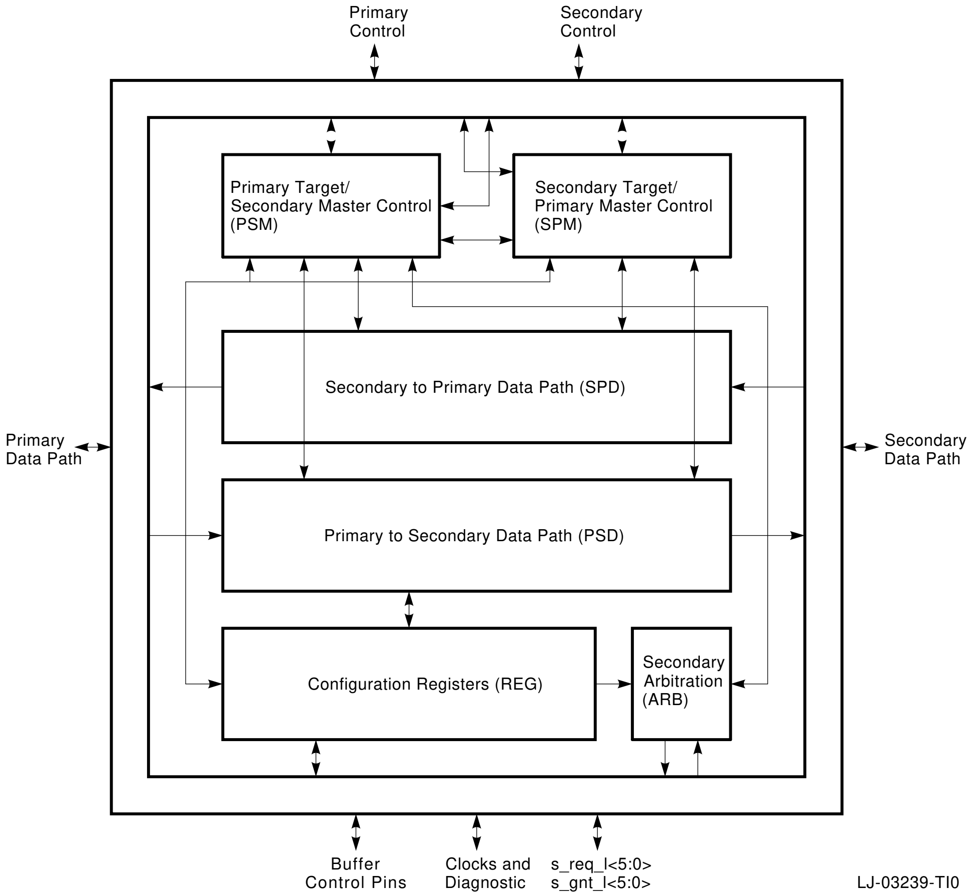


Table 1–1 Major Sub-Blocks of the DECchip 21050

Block	Description
PSM	Machine and control logic for all transactions initiated on the primary interface, whether the transaction is intended for the 21050 itself or a target on the secondary side of the 21050.
SPM	Machine and control logic for all transactions initiated on the secondary interface. All such transactions are intended for a target on the primary interface, since 21050 registers are not accessible from the secondary interface.
SPD	Data path for data received on the secondary interface and driven on the primary interface. Used for writes initiated on the secondary PCI bus or reads initiated on the primary PCI bus.
PSD	Data path for data received on the primary interface and driven on the secondary interface. Used for writes initiated on the primary PCI bus or reads initiated on the secondary PCI bus.
REG	Configuration registers and corresponding control logic. Accessible from the primary interface only.
ARB	Logic for secondary bus arbitration. Receives **s_req_l<5:0>**, as well as the PPB secondary bus request, and drives one of the **s_gnt_l<5:0>** lines or the PPB secondary bus grant.

2

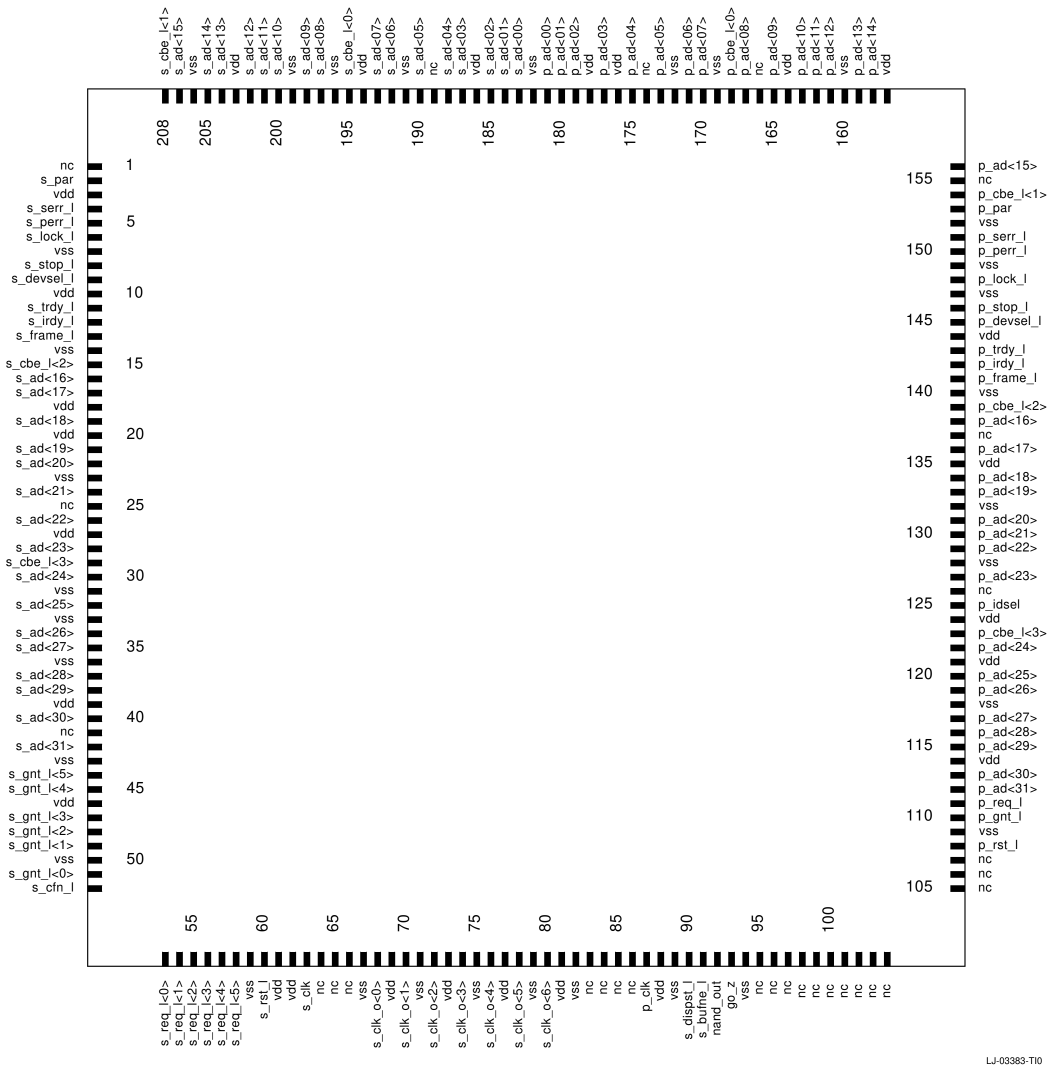
DECchip 21050 Pin Assignment

The 21050 is divided into three main functions:

- Primary PCI bus interface that interacts with the bus closest to the CPU
- Secondary PCI bus interface that interacts with the bus that is farther from the CPU
- Power supply and miscellaneous functions

Figure 2–1 shows the 21050 signals, and the 21050 pin assignments are listed in alphabetic and numeric order at the end of this chapter.

Figure 2–1 DECchip 21050 Pinout Diagram



2.1 Signal Types

The following table defines the 21050 signal types referred to in this chapter.

Signal Type	Definition
I	Standard input only.
O	Standard output only.
ts	Tristate bidirectional.
sts	Sustained tristate. Active low signal must be pulled high for one cycle when deasserting.
od	Standard open drain.
P	Power or ground.

2.2 Alphabetic 21050 Pin Assignment List

Table 2–1 lists the 21050 pins in alphabetic order.

Table 2–1 Alphabetic Pin Assignment List

Pin*	Pin Number	Type	Pin*	Pin Number	Type
go_z	93	I	**nc**	137	–
nand_out	92	O	**nc**	155	–
nc	1	–	**nc**	166	–
nc	25	–	**nc**	174	–
nc	41	–	**nc**	189	–
nc	64	–	**p_ad<00>**	181	ts
nc	65	–	**p_ad<01>**	180	ts
nc	66	–	**p_ad<02>**	179	ts
nc	83	–	**p_ad<03>**	177	ts
nc	84	–	**p_ad<04>**	175	ts
nc	85	–	**p_ad<05>**	173	ts
nc	86	–	**p_ad<06>**	171	ts
nc	95	–	**p_ad<07>**	170	ts
nc	96	–	**p_ad<08>**	167	ts
nc	97	–	**p_ad<09>**	165	ts
nc	98	–	**p_ad<10>**	163	ts
nc	99	–	**p_ad<11>**	162	ts
nc	100	–	**p_ad<12>**	161	ts
nc	101	–	**p_ad<13>**	159	ts
nc	102	–	**p_ad<14>**	158	ts
nc	103	–	**p_ad<15>**	156	ts
nc	104	–	**p_ad<16>**	138	ts
nc	105	–	**p_ad<17>**	136	ts
nc	106	–	**p_ad<18>**	134	ts
nc	107	–	**p_ad<19>**	133	ts
nc	126	–	**p_ad<20>**	131	ts

****nc**—Do not connect these pins on the board.*

Pin	Pin Number	Type	Pin	Pin Number	Type
p_ad<21>	130	ts	**s_ad<00>**	183	ts
p_ad<22>	129	ts	**s_ad<01>**	184	ts
p_ad<23>	127	ts	**s_ad<02>**	185	ts
p_ad<24>	122	ts	**s_ad<03>**	187	ts
p_ad<25>	120	ts	**s_ad<04>**	188	ts
p_ad<26>	119	ts	**s_ad<05>**	190	ts
p_ad<27>	117	ts	**s_ad<06>**	192	ts
p_ad<28>	116	ts	**s_ad<07>**	193	ts
p_ad<29>	115	ts	**s_ad<08>**	197	ts
p_ad<30>	113	ts	**s_ad<09>**	198	ts
p_ad<31>	112	ts	**s_ad<10>**	200	ts
p_cbe_l<0>	168	ts	**s_ad<11>**	201	ts
p_cbe_l<1>	154	ts	**s_ad<12>**	202	ts
p_cbe_l<2>	139	ts	**s_ad<13>**	204	ts
p_cbe_l<3>	123	ts	**s_ad<14>**	205	ts
p_clk	87	I	**s_ad<15>**	207	ts
p_devsel_l	145	sts	**s_ad<16>**	16	ts
p_frame_l	141	sts	**s_ad<17>**	17	ts
p_gnt_l	110	I	**s_ad<18>**	19	ts
p_idsel	125	I	**s_ad<19>**	21	ts
p_irdy_l	142	sts	**s_ad<20>**	22	ts
p_lock_l	148	sts	**s_ad<21>**	24	ts
p_par	153	ts	**s_ad<22>**	26	ts
p_perr_l	150	sts	**s_ad<23>**	28	ts
p_req_l	111	ts	**s_ad<24>**	30	ts
p_rst_l	108	I	**s_ad<25>**	32	ts
p_serr_l	151	od	**s_ad<26>**	34	ts
p_stop_l	146	sts	**s_ad<27>**	35	ts
p_trdy_l	143	sts	**s_ad<28>**	37	ts

Pin	Pin Number	Type	Pin	Pin Number	Type
s_ad<29>	38	ts	**s_req_l<0>**	53	I
s_ad<30>	40	ts	**s_req_l<1>**	54	I
s_ad<31>	42	ts	**s_req_l<2>**	55	I
s_bufne_l	91	O	**s_req_l<3>**	56	I
s_cbe_l<0>	195	ts	**s_req_l<4>**	57	I
s_cbe_l<1>	208	ts	**s_req_l<5>**	58	I
s_cbe_l<2>	15	ts	**s_rst_l**	60	O
s_cbe_l<3>	29	ts	**s_serr_l**	4	I
s_cfn_l	52	I	**s_stop_l**	8	sts
s_clk	63	I	**s_trdy_l**	11	sts
s_clk_o<0>	68	O	**vdd**	20	P
s_clk_o<1>	70	O	**vdd**	61	P
s_clk_o<2>	72	O	**vdd**	124	P
s_clk_o<3>	74	O	**vdd**	176	P
s_clk_o<4>	76	O	**vdd**	3	P
s_clk_o<5>	78	O	**vdd**	10	P
s_clk_o<6>	80	O	**vdd**	18	P
s_devsel_l	9	sts	**vdd**	27	P
s_dispst_l	90	I	**vdd**	39	P
s_frame_l	13	sts	**vdd**	46	P
s_gnt_l<5>	44	ts	**vdd**	62	P
s_gnt_l<4>	45	ts	**vdd**	69	P
s_gnt_l<3>	47	ts	**vdd**	73	P
s_gnt_l<2>	48	ts	**vdd**	77	P
s_gnt_l<1>	49	ts	**vdd**	81	P
s_gnt_l<0>	51	ts	**vdd**	88	P
s_irdy_l	12	sts	**vdd**	114	P
s_lock_l	6	sts	**vdd**	121	P
s_par	2	ts	**vdd**	135	P
s_perr_l	5	sts	**vdd**	144	P

Pin	Pin Number	Type	Pin	Pin Number	Type
vdd	157	P	**vss**	132	P
vdd	164	P	**vss**	140	P
vdd	178	P	**vss**	147	P
vdd	186	P	**vss**	152	P
vdd	194	P	**vss**	160	P
vdd	203	P	**vss**	169	P
vss	33	P	**vss**	172	P
vss	89	P	**vss**	182	P
vss	149	P	**vss**	191	P
vss	196	P	**vss**	199	P
vss	7	P	**vss**	206	P
vss	14	P			
vss	23	P			
vss	31	P			
vss	36	P			
vss	43	P			
vss	50	P			
vss	59	P			
vss	67	P			
vss	71	P			
vss	75	P			
vss	79	P			
vss	82	P			
vss	94	P			
vss	109	P			
vss	118	P			
vss	128	P			

2.3 Numeric 21050 Pin Assignment List

Table 2–2 lists the 21050 pins in numeric order.

Table 2–2 Numeric Pin Assignment List

Pin*	Pin Number	Type	Pin*	Pin Number	Type
nc	1	–	**s_ad<22>**	26	ts
s_par	2	ts	**vdd**	27	P
vdd	3	P	**s_ad<23>**	28	ts
s_serr_l	4	I	**s_cbe_l<3>**	29	ts
s_perr_l	5	sts	**s_ad<24>**	30	ts
s_lock_l	6	sts	**vss**	31	P
vss	7	P	**s_ad<25>**	32	ts
s_stop_l	8	sts	**vss**	33	P
s_devsel_l	9	sts	**s_ad<26>**	34	ts
vdd	10	P	**s_ad<27>**	35	ts
s_trdy_l	11	sts	**vss**	36	P
s_irdy_l	12	sts	**s_ad<28>**	37	ts
s_frame_l	13	sts	**s_ad<29>**	38	ts
vss	14	P	**vdd**	39	P
s_cbe_l<2>	15	ts	**s_ad<30>**	40	ts
s_ad<16>	16	ts	**nc**	41	–
s_ad<17>	17	ts	**s_ad<31>**	42	ts
vdd	18	P	**vss**	43	P
s_ad<18>	19	ts	**s_gnt_l<5>**	44	ts
vdd	20	P	**s_gnt_l<4>**	45	ts
s_ad<19>	21	ts	**vdd**	46	P
s_ad<20>	22	ts	**s_gnt_l<3>**	47	ts
vss	23	P	**s_gnt_l<2>**	48	ts
s_ad<21>	24	ts	**s_gnt_l<1>**	49	ts
nc	25	–	**vss**	50	P

***nc**—Do not connect these pins on the board.

Pin*	Pin Number	Type	Pin*	Pin Number	Type
s_gnt_l<0>	51	ts	**vss**	79	P
s_cfn_l	52	I	**s_clk_o**<6>	80	O
s_req_l<0>	53	ts	**vdd**	81	P
s_req_l<1>	54	I	**vss**	82	P
s_req_l<2>	55	I	**nc**	83	–
s_req_l<3>	56	I	**nc**	84	–
s_req_l<4>	57	I	**nc**	85	–
s_req_l<5>	58	I	**nc**	86	–
vss	59	P	**p_clk**	87	I
s_rst_l	60	O	**vdd**	88	P
vdd	61	P	**vss**	89	P
vdd	62	P	**s_dispst_l**	90	I
s_clk	63	I	**s_bufne_l**	91	O
nc	64	–	**nand_out**	92	O
nc	65	–	**go_z**	93	I
nc	66	–	**vss**	94	P
vss	67	P	**nc**	95	–
s_clk_o<0>	68	O	**nc**	96	–
vdd	69	O	**nc**	97	–
s_clk_o<1>	70	O	**nc**	98	–
vss	71	P	**nc**	99	–
s_clk_o<2>	72	O	**nc**	100	–
vdd	73	P	**nc**	101	–
s_clk_o<3>	74	O	**nc**	102	–
vss	75	P	**nc**	103	–
s_clk_o<4>	76	O	**nc**	104	–
vdd	77	P	**nc**	105	–
s_clk_o<5>	78	O	**nc**	106	–

***nc**—Do not connect these pins on the board.

Pin*	Pin Number	Type	Pin*	Pin Number	Type
nc	107	–	**p_ad<16>**	138	ts
p_rst_l	108	I	**p_cbe_l<2>**	139	ts
vss	109	P	**vss**	140	P
p_gnt_l	110	I	**p_frame_l**	141	sts
p_req_l	111	ts	**p_irdy_l**	142	sts
p_ad<31>	112	ts	**p_trdy_l**	143	sts
p_ad<30>	113	ts	**vdd**	144	P
vdd	114	P	**p_devsel_l**	145	sts
p_ad<29>	115	ts	**p_stop_l**	146	sts
p_ad<28>	116	ts	**vss**	147	P
p_ad<27>	117	ts	**p_lock_l**	148	sts
vss	118	P	**vss**	149	P
p_ad<26>	119	ts	**p_perr_l**	150	sts
p_ad<25>	120	ts	**p_serr_l**	151	od
vdd	121	P	**vss**	152	P
p_ad<24>	122	ts	**p_par**	153	ts
p_cbe_l<3>	123	ts	**p_cbe_l<1>**	154	ts
vdd	124	P	**nc**	155	–
p_idsel	125	ts	**p_ad<15>**	156	ts
nc	126	–	**vdd**	157	P
p_ad<23>	127	ts	**p_ad<14>**	158	ts
vss	128	P	**p_ad<13>**	159	ts
p_ad<22>	129	ts	**vss**	160	P
p_ad<21>	130	ts	**p_ad<12>**	161	ts
p_ad<20>	131	ts	**p_ad<11>**	162	ts
vss	132	P	**p_ad<10>**	163	ts
p_ad<19>	133	ts	**vdd**	164	P
p_ad<18>	134	ts	**p_ad<09>**	165	ts
vdd	135	P	**nc**	166	–
p_ad<17>	136	ts	**p_ad<08>**	167	ts
nc	137	P	**p_cbe_l<0>**	168	ts

*__nc__—Do not connect these pins on the board.

Pin*	Pin Number	Type	Pin*	Pin Number	Type
vss	169	P	**vss**	196	P
p_ad<07>	170	ts	**s_ad<08>**	197	ts
p_ad<06>	171	ts	**s_ad<09>**	198	ts
vss	172	P	**vss**	199	P
p_ad<05>	173	ts	**s_ad<10>**	200	ts
nc	174	–	**s_ad<11>**	201	ts
p_ad<04>	175	ts	**s_ad<12>**	202	ts
vdd	176	P	**vdd**	203	P
p_ad<03>	177	ts	**s_ad<13>**	204	ts
vdd	178	P	**s_ad<14>**	205	ts
p_ad<02>	179	ts	**vss**	206	P
p_ad<01>	180	ts	**s_ad<15>**	207	ts
p_ad<00>	181	ts	**s_cbe_l<1>**	208	ts
vss	182	P			
s_ad<00>	183	ts			
s_ad<01>	184	ts			
s_ad<02>	185	ts			
vdd	186	P			
s_ad<03>	187	ts			
s_ad<04>	188	ts			
nc	189	–			
s_ad<05>	190	ts			
vss	191	P			
s_ad<06>	192	ts			
s_ad<07>	193	ts			
vdd	194	P			
s_cbe_l<0>	195	ts			

*__nc__—Do not connect these pins on the board.

3 Signal Description

This chapter contains a detailed description of 21050 signals. Signals are divided into four major functions:

- Primary PCI bus
- Secondary PCI bus
- Secondary bus arbiter
- Clocks, reset, and miscellaneous

Note

The **_l** symbol at the end of a signal name indicates that the active, or asserted state occurs when the signal is at a low-voltage level. If the **_l** symbol is not present after the signal name, then the signal is asserted at the high voltage level.

The following table describes the signal types referred to in this chapter.

Signal Type	Description
I	Standard input only.
O	Standard output only.
ts	Tristate bidirectional.
sts	Sustained tristate. Active low signal must be pulled high for one cycle when deasserting.
od	Standard open drain.

3.1 Primary PCI Bus Signals

The following table describes the primary PCI bus signals.

Signal Name	Type	Description
p_ad<31:0>	ts	**Primary PCI interface address/data**. These signals represent a multiplexed PCI address and data bus. During an address phase of a transaction, **p_ad<31:0>** contains a physical byte address. During subsequent data phases, **p_ad<31:0>** contains data. A PCI bus transaction consists of one or two address phases followed by one or more data phases.
p_cbe_l<3:0>	ts	**Primary PCI interface command/byte enables**. These signals are multiplexed bus command and byte enables. During an address phase of a transaction, **p_cbe_l<3:0>** contains the bus command that defines the type of PCI transaction. During data phases, **p_cbe_l<3:0>** contains byte enables dictating which byte lanes carry valid data. **p_cbe_l<0>** applies to byte 0; **p_cbe_l<3>** applies to byte 3.
p_frame_l	sts	**Primary PCI interface cycle frame**. **p_frame_l** is driven by the initiator of the transaction to indicate the beginning and duration of an access on the primary PCI bus. **p_frame_l** assertion indicates the beginning of an access. While **p_frame_l** is asserted, data transfers continue. The deassertion of **p_frame_l** indicates the final data phase. The bridge samples **p_frame_l** as an input and also drives **p_frame_l** when acting as the initiator of a transaction on the primary PCI bus.
p_trdy_l	sts	**Primary PCI interface target ready**. This signal indicates the target's ability to complete the current data phase of a transaction on the primary PCI bus. The bridge drives **p_trdy_l** when acting as a target on the primary PCI bus and samples **p_trdy_l** when acting as an initiator on the primary PCI bus.
p_irdy_l	sts	**Primary PCI interface initiator ready**. This signal indicates the initiator's ability to complete the current data phase of a transaction on the primary PCI bus. The bridge drives **p_irdy_l** when acting as an initiator on the primary PCI bus and samples **p_irdy_l** when acting as a target on the primary PCI bus.

Signal Name	Type	Description
p_stop_l	sts	**Primary PCI interface stop indicator**. This signal indicates that the current target is requesting the bus initiator to stop the current transaction on the primary PCI bus. The bridge drives **p_stop_l** when acting as a target on the primary PCI bus and samples **p_stop_l** when acting as an initiator on the primary PCI bus.
p_lock_l	sts	**Primary PCI interface resource lock**. Indicates an atomic operation that may require multiple transactions to complete. Resources in the bridge cannot be locked, but the bridge does propagate locks across the bridge. The bridge samples **p_lock_l** when acting as a target on the primary PCI bus and drives **p_lock_l** when acting as an initiator on the primary PCI bus on behalf of a master on the secondary bus.
p_idsel	I	**Initialization device select**. Used as a chip select during configuration read and write commands. If **p_idsel** is detected asserted and the transaction is a Type 0 configuration command, then the bridge responds as a target to the transaction by asserting **p_devsel_l**.
p_devsel_l	sts	**Primary PCI interface device select.** Asserted by the bridge through positive decoding of the address on **p_ad<31:0>**. When it is accepting a transaction for an internal configuration access or when it is forwarding a transaction across the bridge. The bridge samples **p_devsel_l** when it is acting as an initiator on the primary PCI bus, and expects **p_devsel_l** to be asserted within five cycles of **p_frame_l** assertion. Otherwise, the transaction is terminated with an master abort.
p_par	ts	**Primary PCI interface parity.** Even parity, calculated on 36 bits composed of **p_ad<31:0>** and **p_cbe_l<3:0>**. The **p_par** signal is generated for all address and data phases and is valid one clock cycle after valid data or address is driven on **p_ad<31:0>**. The **p_par** signal is driven and tristated identically to **p_ad**, except that it is delayed one clock cycle. The **p_par** signal is driven by the bridge when acting as an initiator during address phases and write data phases. The **p_par** signal is driven by the bridge when acting as a target during read data phases. The **p_par** signal is sampled as an input during all address phases, and when acting as a target during write data phases.

Signal Name	Type	Description
p_serr_l	od	**Primary PCI interface system error**. Can be pulsed by any device residing on the primary PCI bus that detects a system error condition. The bridge can be enabled to assert **p_serr_l** as a result of address parity error detection, special cycle data parity error, **s_serr_l** assertion, master aborts or target aborts during posted writes, data parity error during posted writes, and undelivered posted write data.
p_perr_l	sts	**Primary PCI interface parity error detected**. Asserted when a data parity error is detected, and corresponds to **p_par** driven one clock cycle earlier. The bridge asserts **p_perr_l** when it detects a write data parity error when acting as a target, or a read data parity error when acting as an initiator.
p_req_l	ts	**Primary PCI bus request**. Asserted by the bridge to indicate to the bus arbiter that it wants to use the primary PCI bus.
p_gnt_l	I	**Primary PCI bus grant**. When asserted, indicates to the bridge that access to the primary PCI bus is granted. The bridge can start a transaction as soon as **p_gnt_l** is asserted and the bus is idle.

3.2 Secondary PCI Bus Signals

The following table describes the secondary PCI bus signals.

Signal Name	Type	Description
s_ad<31:0>	ts	**Secondary PCI interface address/data**. These signals represent a multiplexed PCI address and data bus. During an address phase of a transaction, **s_ad<31:0>** contains a physical byte address. During subsequent data phases, **s_ad<31:0>** contains data. A PCI-to-PCI bridge transaction consists of one or two address phases followed by one or more data phases.
s_cbe_l<3:0>	ts	**Secondary PCI interface command/byte enables**. These signals represent multiplexed bus command and byte enables. During an address phase of a transaction, **s_cbe_l<3:0>** contains the bus command defining the type of PCI transaction. During data phases, **s_cbe_l<3:0>** contains byte enables dictating which byte lanes carry valid data. **s_cbe_l<0>** applies to byte 0; **s_cbe_l<3>** applies to byte 3.

Signal Name	Type	Description
s_frame_l	sts	**Secondary PCI interface cycle frame**. Driven by the initiator of the transaction to indicate the beginning and duration of an access on the secondary PCI bus. The assertion of **s_frame_l** indicates the beginning of an access. While **s_frame_l** is asserted, data transfers continue. The deassertion of **s_frame_l** indicates the final data phase. The bridge samples **s_frame_l** as an input and also drives **s_frame_l** when acting as the initiator of a transaction on the secondary PCI bus.
s_trdy_l	sts	**Secondary PCI interface target ready**. This signal indicates the target agent's ability to complete the current data phase of a transaction on the secondary PCI bus. The bridge drives **s_trdy_l** when acting as a target on the secondary PCI bus and samples **s_trdy_l** when acting as an initiator on the secondary PCI bus.
s_irdy_l	sts	**Secondary PCI interface initiator ready**. This signal indicates the initiator's ability to complete the current data phase of a transaction on the secondary PCI bus. The bridge drives **s_irdy_l** when acting as an initiator on the secondary PCI bus and samples **s_irdy_l** when acting as a target on the secondary PCI bus.
s_stop_l	sts	**Secondary PCI interface stop indicator**. This signal indicates that the current target is requesting the bus initiator to stop the current transaction on the secondary PCI bus. The bridge drives **s_stop_l** when acting as a target on the secondary PCI bus and samples **s_stop_l** when acting as an initiator on the secondary PCI bus.
s_lock_l	sts	**Secondary PCI interface resource lock**. Indicates an atomic operation that may require multiple transactions to complete. The bridge cannot be locked, but it does propagate locks across the bridge. The bridge samples **s_lock_l** when acting as a target on the secondary PCI bus and may drive **s_lock_l** when acting as an initiator on the secondary PCI bus on behalf of a master on the primary bus.
s_devsel_l	sts	**Secondary PCI interface device select.** Asserted by the bridge through positive decoding of the address on **s_ad<31:0>**. When it is forwarding a transaction upstream across the bridge. The bridge samples **s_devsel_l** when it is acting as an initiator on the secondary PCI bus, and expects **s_devsel_l** to be asserted within five cycles of **s_frame_l** assertion. Otherwise, the transaction is terminated with a master abort.

Signal Name	Type	Description
s_par	ts	**Secondary PCI interface parity**. Even parity calculated on 36 bits composed of **s_ad<31:0>** and **s_cbe_l<3:0>**. The **s_par** signal is generated for all address and data phases and is valid one clock cycle after valid data or address is driven on **s_ad**. The **s_par** signal is driven and tristated identically to **s_ad**, except that it is delayed clock cycle. The **s_par** signal is driven by the bridge when acting as an initiator during address phases and write data phases. The **s_par** signal is driven by the bridge when acting as a target during read data phases. The **s_par** signal is sampled as an input during all address phases, and when acting as a target during write data phases.
s_serr_l	I	**Secondary PCI interface system error**. Can be pulsed by any device residing on the secondary PCI bus that detects a system error condition. The bridge does not assert **s_serr_l** as an output. The bridge can be enabled to detect assertion of **s_serr_l** as an input and cause **p_serr_l** to assert as a result.
s_perr_l	sts	**Secondary PCI interface parity error detected**. Asserted when a data parity error is detected, and corresponds to **s_par** driven one clock cycle earlier. The bridge asserts **s_perr_l** when it detects a write data parity error when acting as a target, or a read data parity error when acting as an initiator.

3.3 Secondary Bus Arbiter Signals

The following table describes the secondary bus arbiter signals.

Signal Name	Type	Description
s_req_l<5:0>	I	**Secondary PCI bus request inputs**. The bridge accepts up to six secondary bus inputs to its secondary bus arbiter. The bridge input to the arbiter is an internal chip signal. An asserted level on one of the **s_req_l<5:0>** pins indicates that an initiator wishes to use the secondary PCI bus. If the internal arbiter is disabled, then the **s_req_l<0>** input is reconfigured to be an external secondary bus grant input. In this case, an asserted level on **s_req_l<0>** indicates that the bridge can start an access on the secondary bus.
s_gnt_l<5:0>	ts	**Secondary PCI bus grant outputs**. The bridge secondary bus arbiter can assert one of the **s_gnt_l<5:0>** outputs to indicate that an initiator can start an access the secondary PCI bus if the bus is idle. Only one **s_gnt_l<5:0>** signal can be asserted during any given clock cycle. The bridge bus grant is an internal chip signal. The arbiter can be configured to be in rotating mode, or in a dual alternating/rotating mode. If the internal arbiter is disabled, then the **s_gnt_l<0>** output is reconfigured to be the bridge external secondary bus request. The bridge will then assert this signal whenever it wants to use the secondary PCI bus.
s_cfn_l	I	**Secondary central function enable.** When low, the on-chip secondary bus arbiter is enabled. Also, the secondary PCI bus will be parked at the bridge when the bus is idle. When high, the on-chip secondary bus arbiter is disabled and an external arbiter should be used. The **s_gnt_l<0>** input is reconfigured to be the bridge’s external secondary bus request, and the **s_req_l<0>** input is reconfigured to be the bridge’s external secondary bus grant. The external arbiter must also park the secondary bus, and can park the bus at the bridge by asserting **s_req_l<0>** when the secondary bus is idle.

3.4 Clock, Reset, and Miscellaneous Signals

The following table describes the clock, reset, and miscellaneous signals.

Signal Name	Type	Description
p_clk	I	**Primary PCI bus clock input**. Provides timing for all transactions on the primary PCI bus. All primary PCI bus inputs are sampled on the rising edge of **p_clk**, and all primary PCI bus outputs are driven from the rising edge of **p_clk**. Frequencies supported by the bridge range from 0 to 33 megahertz.
p_rst_l	I	**Primary PCI bus reset**. Forces the bridge to a known state. All register state is cleared and all primary PCI bus outputs are tristated. The **p_rst_l** signal may be asynchronous with **p_clk**. The **p_rst_l** signal must be asserted for at least ten PCI clock cycles in order to reset the bridge properly.
s_clk	I	**Secondary PCI bus clock input**. Provides timing for all transactions on the secondary PCI bus. All secondary PCI bus inputs, as well as miscellaneous inputs with **s_** prefixes, are sampled on the rising edge of **s_clk**; all secondary PCI bus outputs, as well as miscellaneous outputs with **s_** prefixes, are driven from the rising edge of **s_clk**. Frequencies supported by the bridge range from 0 to 33 megahertz.
s_clk_o<6:0>	O	**Secondary PCI bus clock outputs**. Clock outputs are buffered versions of **p_clk** and may be used as secondary PCI bus clocks. If used, one of these clock outputs must be used as a bridge **s_clk** input. The **s_clk_o<6:0>** clock may not be used to drive connector slots, and may not be used for the **p_clk** input of another 21050.

Signal Name	Type	Description
s_rst_l	O	**Secondary PCI bus reset**. Asserted by the bridge under any of the following conditions: • The **p_rst_l** signal is asserted. • The secondary reset bit is set. • The chip reset bit is set. When the bridge asserts **s_rst_l**, it tristates all secondary PCI control signals, and drives zeros on **s_ad**, **s_cbe_l**, and **s_par**. The **s_rst_l** signal remains asserted until **p_rst_l** is deasserted, or the secondary reset bit is cleared. Assertion of **s_rst_l** does not clear the bridge register state, and bridge configuration registers are still accessible from the primary interface.
s_dispst_l	I	**Disable posting control**. When the bridge detects this input asserted, the bridge asserts **s_bufne_l** if any write data exists in data buffers. If the bridge is configured to disable posting upon **s_dispst_l** assertion, no write data is posted and write transactions are limited to single burst transfers until **s_dispst_l** is deasserted. This signal should be tied high if no external disable posting control is required.
s_bufne_l	O	**Buffer not empty indication**. When the bridge asserts this output, write data stored in bridge data buffers at the time of **s_dispst_l** assertion is not yet flushed. When deasserted, any write data in data buffers at the time of **s_dispst_l** assertion is flushed.
goz_l	I	**Diagnostic tristate control**. When asserted, tristates all bidirectional and tristateable output pins.
nand_out	O	**Nand tree diagnostic output**. This pin is dedicated to the diagnostic Nand tree. The Nand tree starts at **s_dispst_l** and runs counter-clockwise. All inputs, except **p_clk** and **s_clk**, are used in the Nand tree. The **goz_l** signal should be asserted when the Nand tree feature is used.

4
Functional Description

This chapter provides a functional description of the 21050 including:

- Bus interfaces and arbitration
- Transaction types
- Latency control, exclusive access, and error handling

4.1 PCI Bus Interfaces

The 21050 bridges system traffic between the primary PCI bus or the PCI bus closer to the CPU, and the secondary PCI bus or the bus farther from the CPU.

4.1.1 Primary PCI Bus Interface

The 21050 can act as either initiator or target on the primary PCI bus. Primary PCI bus signal names are prefaced with ***p_***.

The primary PCI interface consists of the standard set of PCI signals. No sideband signals are defined.

4.1.2 Secondary PCI Bus Interface

Secondary PCI bus signal names are prefaced with ***s_***.

In addition to the standard PCI bus signals, the secondary PCI interface implements additional arbiter, data synchronization, and clock signals. The use of these signals is not required for proper functionality. The secondary interface has neither an idsel input or a reset input.

The secondary interface has one input, **s_dispst_l**, for write buffer flushing and disabling of write posting. A secondary interface output, **s_bufne_l**, indicates when data in write buffers at the time of **s_dispst_l** assertion are emptied.

The secondary PCI interface accepts a separate clock signal for driving and sampling the secondary data and control signals. The 21050 assumes that the secondary clock is a buffered version of the primary clock. For a description of the skew specification between **p_clk** and **s_clk**, see Section 4.13.

The 21050 also generates seven buffered versions of **p_clk**, called **s_clk_o<6:0>**, which may be used as secondary interface clocks, one of which may be connected to **s_clk**.

The secondary interface has a reset output, **s_rst_l**, that the 21050 drives when it detects **p_rst_l** asserted or the secondary reset control bit is set, or when the chip reset bit is set.

Configuration accesses forwarded to the secondary bus use the upper **s_ad** pins during the address cycle as outgoing secondary idsel signals. The secondary interface does not have an idsel input because 21050 configuration space is accessible from the primary interface only.

4.2 PCI Address Phase

The PCI address phase is denoted by a falling edge of ***x*frame_l**. A typical PCI transaction consists of a single address phase that lasts for one PCI clock cycle. An exception to this is the dual address transaction, which contains a 64-bit address driven in two subsequent PCI clock cycles. The falling edge of ***x*frame_l** indicates the first address phase.

During a PCI address phase, the ***x*ad<31:0>** signals carry the address of the transaction, while ***x*cbe_l<3:0>** signals carry the transaction code.

The 21050 supports the following address phase features:

- Linear increment address mode
- Address and data stepping
- Dual address support

4.2.1 Linear Increment Address Mode

The 21050 supports only linear increment address mode. If any other address mode is detected, the 21050 disconnects the transaction after the first burst.

4.2.2 Address and Data Stepping

The 21050 supports address and data stepping by other address and data sources as a target.

The 21050 uses address stepping as a master only when generating a Type 0 configuration address phase on the secondary bus. One address step cycle is needed to allow the secondary bus idsel signal to become valid, since it may be connected to an **s_ad** line through resistive coupling.

When address stepping is used, the secondary bus arbiter may deassert a grant before the assertion of **s_frame_l** but it never asserts another grant in the same PCI cycle as the deassertion when the secondary PCI bus is idle. This prevents address contention when address stepping is used on the secondary interface.

4.2.3 Dual Addressing

Dual-address transactions provide a mechanism to generate a 64-bit address on a 32-bit PCI bus. A dual-address transaction consists of two address phases. The first address phase contains the least-significant 32 bits of the address on ***x*ad<31:0>** and the dual-address command on ***x*cbe_l<3:0>**. The second address phase contains the most-significant 32 bits of the address on ***x*ad<31:0>** and the transaction type command on ***x*cbe_l<3:0>**.

The 21050 responds to all dual-address transactions on the secondary bus, and forwards them to the primary bus. The 21050 ignores all dual-address transactions initiated on the primary bus.

When forwarding a dual-address transaction upstream, the 21050 may post write data or prefetch read data, depending on the transaction type. The 21050 follows the same guidelines for posting and prefetching as described for the specific transaction types.

4.3 Device Select (devsel) Generation

In all transactions crossing the 21050 that require a return of ***x*devsel_l** to the initiator, the 21050 performs positive decoding of the address information and conditionally returns ***x*devsel_l** with medium timing to the initiator of the transaction. This ensures that ***x*devsel_l** is returned to the initiating device within the minimum allowed number of PCI clock cycles (5) after assertion of ***x*frame_l**.

If the 21050 does not receive an asserted value of ***y*devsel_l** from the target device within the required number of PCI cycles after the 21050 asserts ***y*frame_l**, the 21050 performs a master abort on the target bus. The 21050 may be configured to handle master aborts in two different ways based on the Master Abort Mode bit.

In default mode, the 21050 asserts ***x*trdy_l** on the initiator bus and completes the transaction. Write data is not delivered, and read data is driven as FFFFFFFFh. The Received Master Abort bit is set in the status register corresponding to the target bus.

If the Master Abort Mode bit is set, then the 21050 is enabled to perform a target abort on the initiator bus when a master abort is detected on the target bus for read and non-posted write operations. The 21050 asserts **p_serr_l** if a master abort occurs during a posted write, if the primary **serr** driver enable is set and the **serr** disable for master abort is not set. The Received Master Abort bit is set in the status register corresponding to the target bus. For reads and non-posted writes, the Signaled Target Abort bit is set in the status register corresponding to the initiator bus. For posted writes, the Signaled System Error bit is set in the Primary Status Register.

4.4 Transaction Forwarding

The 21050 forwards all types of memory, I/O, and configuration commands, and also generates special cycles.

PCI commands are defined by the ***x*cbe_l** signals during the address phase. Table 4–1 lists the ***x*cbe_l** commands.

Table 4–1 *x*cbe_l Commands

*x*cbe_l	PCI Command	*x*cbe_l	PCI Command
0000	Interrupt acknowledge	1000	Reserved
0001	Special cycle	1001	Reserved
0010	I/O read	1010	Configuration read
0011	I/O write	1011	Configuration write
0100	Reserved	1100	Memory read multiple

(continued on next page)

Table 4–1 (Cont.) *x*cbe_l Commands

*x*cbe_l	Command	*x*cbe_l	Command
0101	Reserved	1101	Reserved
0110	Memory read	1110	Memory read line
0111	Memory write	1111	Memory write and invalidate

4.4.1 Transaction Support

Table 4–2 shows which transactions are supported by the 21050, and whether posting or prefetching is used for a particular transaction under normal operating conditions.

- P → S indicates transactions in which the initiator resides on the primary side and the target resides on the secondary side.
- S → P indicates transactions in which the initiator resides on the secondary side and the target resides on the primary side.

Table 4–2 Transaction Forwarding and Filtering

Transaction	P → S	S → P	Notes
Interrupt acknowledge	No	No	–
Special cycle	No	No	–
I/O read	Yes	Yes	Limited to one data transfer; no prefetching.
I/O write	Yes	Yes	Limited to one data transfer; no posting.
Memory read	Yes	Yes	Allows prefetching and multiple data transfers if addressing the prefetchable memory range; otherwise, limited to one data transfer; no prefetching.
Memory write	Yes	Yes	Posting and multiple data transfers allowed.
Configuration read	Yes	Yes	Upstream: Type 1 → Type 1 only. Limited to one data transfer; no prefetching.

(continued on next page)

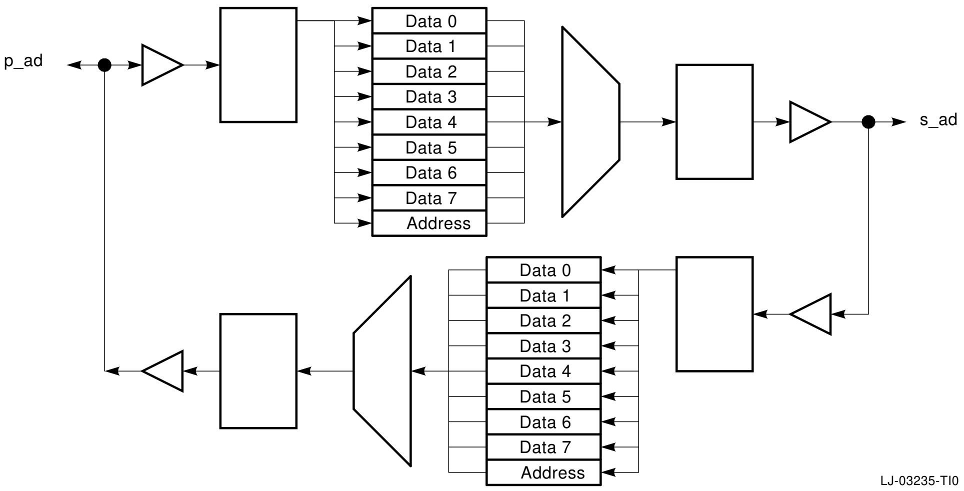
Table 4–2 (Cont.) Transaction Forwarding and Filtering

Transaction	P → S	S → P	Notes
Configuration write	Yes	Yes	Upstream: Type 1 → Type 1 and Type 1 → special cycle only. Limited to one data transfer; no posting.
Memory read multiple	Yes	Yes	Allows prefetching and multiple data transfers with certain restrictions. See Section 4.4.4.4 and Section 4.4.4.5.
Memory read line	Yes	Yes	Allows prefetching and multiple data transfers with certain restrictions. See Section 4.4.4.4 and Section 4.4.4.5.
Memory write and invalidate	Yes	Yes	Posting and multiple data transfers allowed.
Configuration write → special cycle	Yes	Yes	–
Dual address	No	Yes	Posting, prefetching, and burst limitations dependent on transaction type.

4.4.2 Data Path

Figure 4–1 shows the data path for forwarding transactions across the 21050. In each direction eight data buffers are used for either read or write data, and an additional register is used to hold and track addresses.

Figure 4–1 DECchip 21050 Data Path



4.4.3 Write Transactions

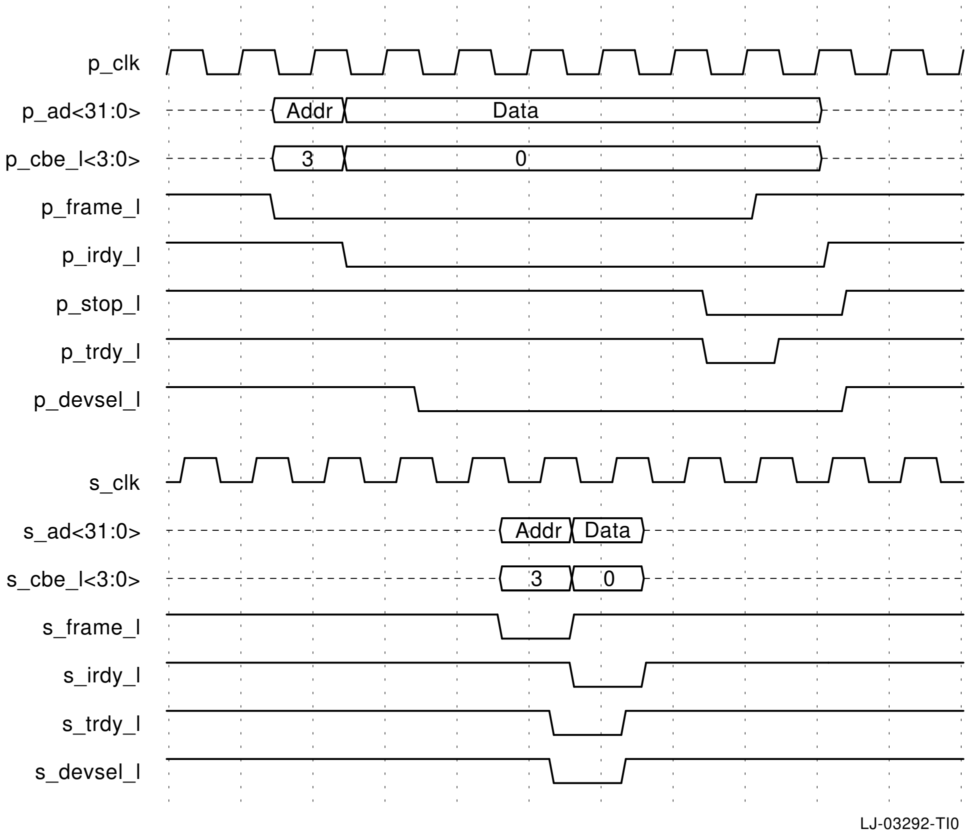
Write transactions may be posted or non-posted. Posted write transactions allow multiple burst data transfers to the 21050 before access to the target device is acquired. Non-posted write transactions are limited to a single data phase. Data is not transferred from the initiator until it is transferred to the target. Posted and non-posted write behavior is described in the following sections.

4.4.3.1 Non-Posted Write Transactions

When a non-posted write that will be forwarded is initiated, the 21050 returns ***x*devsel_l** to the initiator and attempts to gain access to the target bus. After the initiator presents the write data, as indicated by an asserted value of ***x*irdy_l**, and the 21050 obtains possession of the target bus, the 21050 propagates the write data across the 21050 and transfers the data to the target. After data is successfully transferred to the target by assertion of ***y*irdy_l** and ***y*trdy_l**, the 21050 transfers the write data from the initiator by asserting ***x*trdy_l**. Up to this point, the initiator is held in wait states through a target stall. Note that for this type of write, additional latency is incurred since information is propagated in both directions across the 21050. Writes of this type are limited to a single data transfer.

Figure 4–2 shows a non-posted I/O write transaction where the bridge returns a target disconnect after the first data transfer.

Figure 4–2 I/O Write Timing



4.4.3.2 Use of Non-Posted Write Transactions

Non-posted write transactions are used for:

- I/O writes
- Configuration writes
- Memory writes when **s_dispst_l** is asserted and the Posting Disabled bit is set in the Chip Control register

- Memory writes when the target bus is locked, indicated by ***y*****lock_l** asserted.

4.4.3.3 Posted Write Transactions

When a posted write that will be forwarded is initiated, the 21050 returns both ***x*****devsel_l** and ***x*****trdy_0** to the initiator to start accepting write data, before the 21050 acquires access to the target bus. The 21050 attempts to gain access to the target bus in the same manner and with the same timing as for the non-posted write.

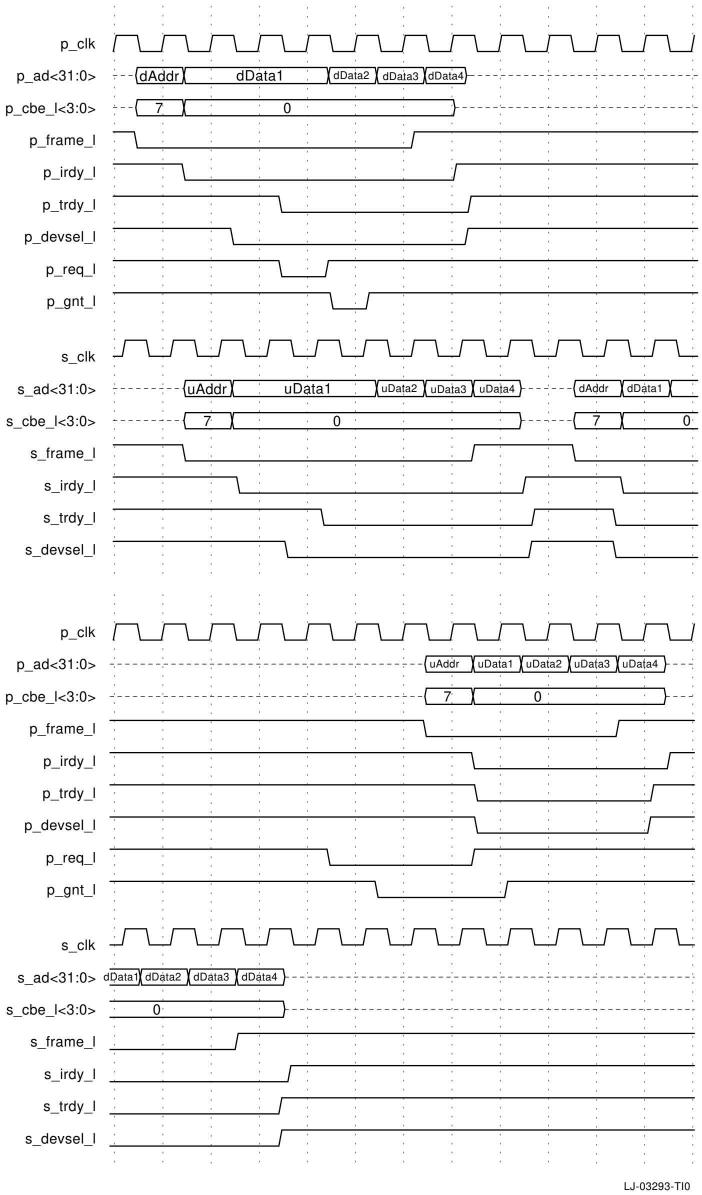
Figure 4–3 shows a posted write transaction. In this diagram, data is posted to both upstream and downstream write buffers simultaneously from both the primary and secondary buses. As soon as the secondary bus becomes idle, the bridge delivers downstream write data to the secondary bus target. When the downstream transaction is complete, upstream write data is delivered to the primary bus target.

For posted write transactions crossing the 21050, the initiator can post up to 8 dwords of data to the 21050 while access to the target bus is acquired. If the 21050 does not receive a target response (an asserted value of ***x*****trdy_l**) by the time the eighth dword is about to be transferred, the 21050 performs a target disconnect on the eighth dword transfer. This is because the 21050 cannot guarantee that the ninth dword will be transferred within eight PCI clock cycles.

If the target returns ***x*****trdy_l** before the write buffer is filled and more than eight dwords are to be transferred, the 21050 allows continuous write data transfers from the initiator until a write boundary is reached. If the target device stalls, causing the write buffer to fill, the 21050 causes a target stall on the initiating bus until buffer space is available.

Write buffering allows posted write transactions to proceed at the maximum transfer rate of one data transfer per PCI clock cycle. The 21050 introduces a delay of two PCI clock cycles between interfaces, but buffering hides the effects this latency would have on throughput.

Figure 4–3 Simultaneous Upstream and Downstream Posted Memory Writes



4.4.3.4 Use of Posted Write Transactions

Posted write transactions are used for:

- Memory write transactions
- Memory write and invalidate transactions

For posted write transactions, **s_dispst_l** must be deasserted or the Disable Posting bit must not be set in the Chip Control register in 21050 configuration space. In addition, the target bus must not be locked, that is, **ylock_l** must be deasserted.

4.4.3.5 Write Boundaries

The 21050 imposes write boundaries on posted write transactions. When the 21050 detects a write boundary, it returns a target disconnect to the initiator and completes delivery of posted write data up to the point of disconnection.

Posted writes are disconnected at cache line boundaries if *both* of the following conditions occur:

- Transaction is a memory write and invalidate
- Cache line size is set to a non-zero value, but is less than sixteen.

Posted writes are disconnected at a 256-dword boundary if *one* of the following conditions occur:

- Transaction is a memory write
- Transaction is a memory write and invalidate, and the cache line size register is 0 or greater than or equal to 16. (Converted to memory write.)

The cache line size is programmable in the Cache Line size register in 21050 configuration space.

4.4.3.6 Conversion of Memory Write and Invalidate Transactions

If the 21050 cannot guarantee transfer of a memory write and invalidate transaction up to a cache line boundary, it converts the transaction to a memory write transaction on the target bus. The 21050 converts a memory write transaction in each of the following situations:

- Cache line size register is set to 0—the 21050 does not know the cache line size.
- Cache line size register is set to 16 dwords or greater—the 21050 is unable to buffer an entire cache line.
- Target has returned a target disconnect and a partial cache line remains in write buffers.

- Cache line size register is set to a value greater than the burst limit counter, and the burst limit counter is non-zero.

The 21050 assumes that the cache line size is a power of 2. If a value that is not a power of 2 is written into the cache line size register, the 21050 assumes the next lower power of 2 number as the cache line size (that is, if a 6 is written, a cache line size of 4 is used).

4.4.3.7 Fast Back-to-Back Write Transactions

The 21050 is capable of responding to fast back-to-back write transactions. The 21050 does not use fast timing for ***x*devsel_l** assertion and the 21050 state machine logic is able to detect the beginning of the second transaction and accurately decode the address. However, if the first transaction is a posted write and both transactions are directed across the 21050, the 21050 returns a target retry in response to the second transaction. This is because the 21050 cannot accept the second transaction until the first transaction is complete on both the primary and secondary interfaces. In the case of a posted write, the transaction completes on the initiator bus before it completes on the target bus. The 21050 can forward back-to-back non-posted writes. The 21050 can also accept fast back-to-back single burst configuration writes to its configuration space.

4.4.4 Read Transactions

Read transactions may prefetch or not prefetch. Reads that do not use prefetching are limited to one data phase and only data requested by the initiator is read. Reads that use prefetching can consist of multiple data phases and can read additional data than that requested by the initiator.

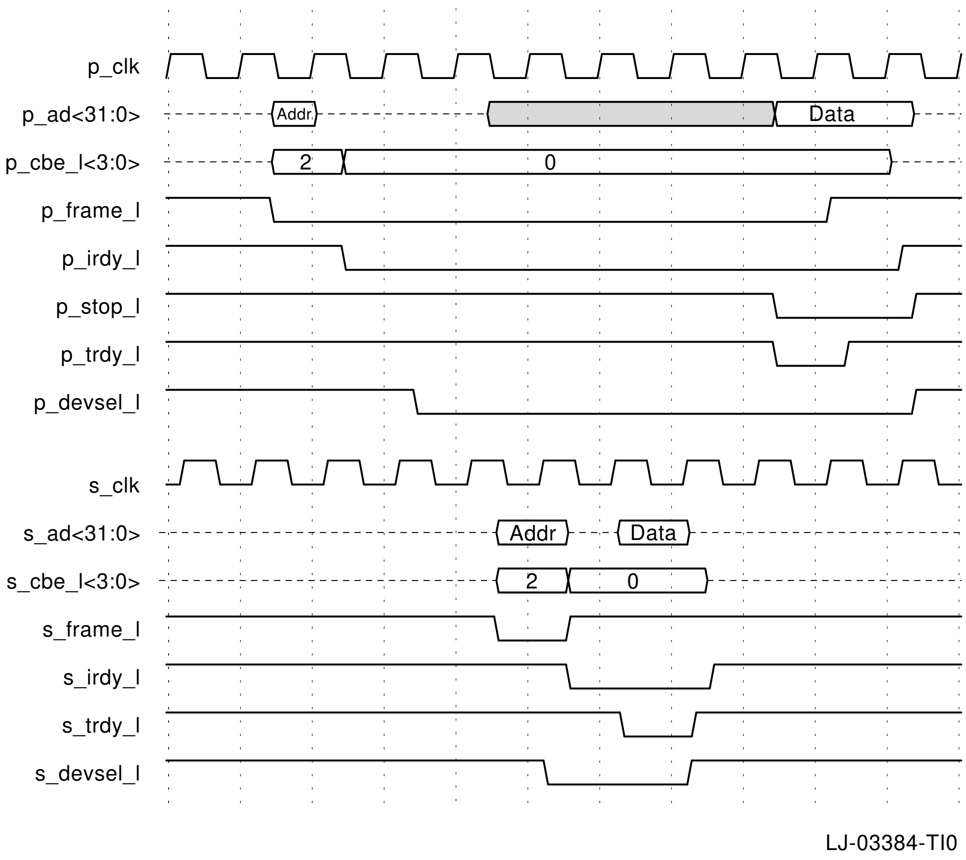
For either type of read that crosses the 21050, possession of both the initiator and target bus, as well as any intermediate PCI buses between the two, must be gained before any data is transferred from the target. Non-prefetchable and prefetchable reads are discussed in the following sections.

4.4.4.1 Non-Prefetchable Reads

When a non-prefetchable read to be forwarded is initiated, the 21050 returns ***x*devsel_l** to the initiator and attempts to gain access to the target bus. When the 21050 initiates the transaction on the target bus, it asserts ***y*frame_l** for only one cycle, indicating a single data transfer request. The 21050 also asserts only those byte enables asserted by the initiator so that nonrequested data is not read. The 21050 holds the initiator in wait states with a target stall until the target device has returned ***y*devsel_l** and ***y*trdy_l** to the 21050, indicating that read data is ready. The 21050 propagates the read data across the 21050, asserting ***x*trdy_l** to transfer it to the initiator. If a multiple burst transfer is requested, then the 21050 also asserts ***x*stop_l**, indicating that no more data will be transferred.

Figure 4–4 shows a non-prefetchable I/O read that is disconnected after the first data transfer.

Figure 4–4 I/O Read Timing



4.4.4.2 The Use of Non-Prefetchable Reads

The 21050 treats the following transactions as non-prefetchable:

- I/O read
- Configuration read
- Memory read and the read address for a downstream read falls into the non-prefetchable memory address range (Section 4.6)
- Upstream memory read and the prefetch disable bit is set in the Chip Control register in 21050 configuration space.

In addition, the 21050 never prefetches data for any type of read where only a single dword is requested and the initiator is not held in wait state (that is, when ***x*frame_l** is asserted for only one cycle).

4.4.4.3 Prefetchable Reads

The protocol for target access is similar to that for non-prefetchable reads. When a prefetchable read to be forwarded is initiated, the 21050 returns ***x*devsel_l** to the initiator and attempts to gain access to the target bus. When the 21050 gains access to the target bus and begins the read transaction, it asserts all byte enables for all data transfers, regardless of which byte enables are asserted by the initiator. The 21050 holds the initiator in wait states by a target stall until the target device has returned ***y*devsel_l** and ***y*trdy_l** to the 21050, indicating that read data is ready. The 21050 propagates the read data across the 21050, asserting ***x*trdy_l** to transfer the data to the initiator.

In prefetchable read (unlike a non-prefetchable read), the 21050 keeps ***y*frame_l** asserted on the target bus until either a read boundary is reached or until ***x*frame_l** deasserts on the target bus, whichever comes first.

In the case of ***x*frame_l** deassertion, up to four extra dwords that are not requested by the initiator due to 21050 latency, may be read from the target. These extra dwords are discarded. If a read boundary is encountered before the initiator deasserts ***x*frame_l**, no nonrequested dwords are read.

Figure 4–5 and Figure 4–6 show prefetchable reads.

Figure 4–5 shows a read up to a cache line boundary. If the initiator had requested more data, it would have been disconnected by the 21050.

Figure 4–6 shows how extra dwords may be read due to the effects of bridge latency when forwarding the ***x*frame_l** signal.

Figure 4–5 Memory Read with Prefetching up to Cache Line Boundary

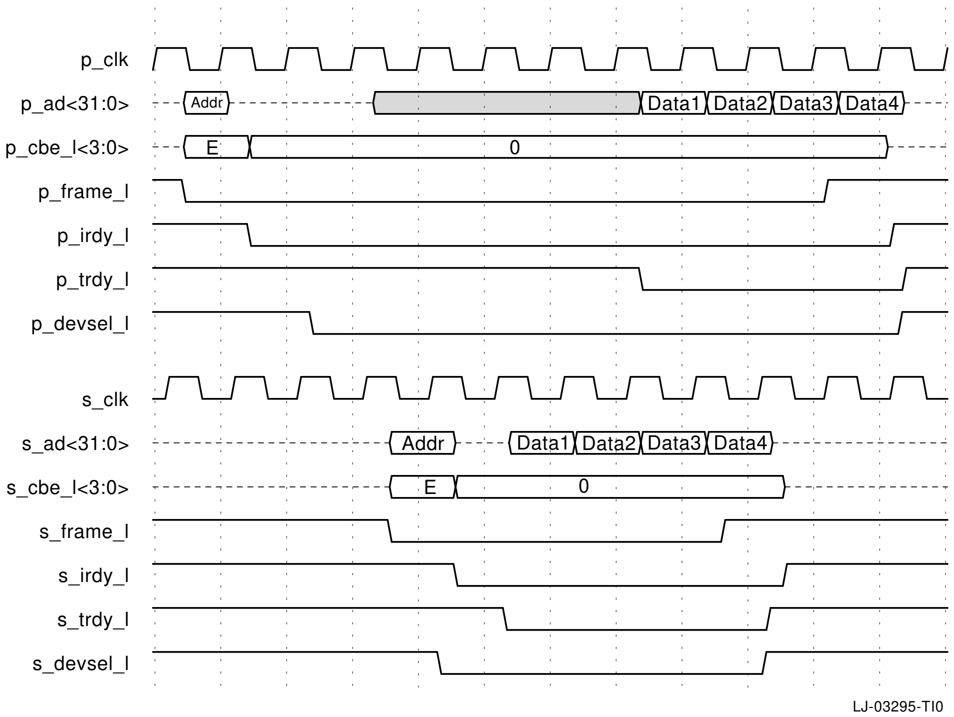
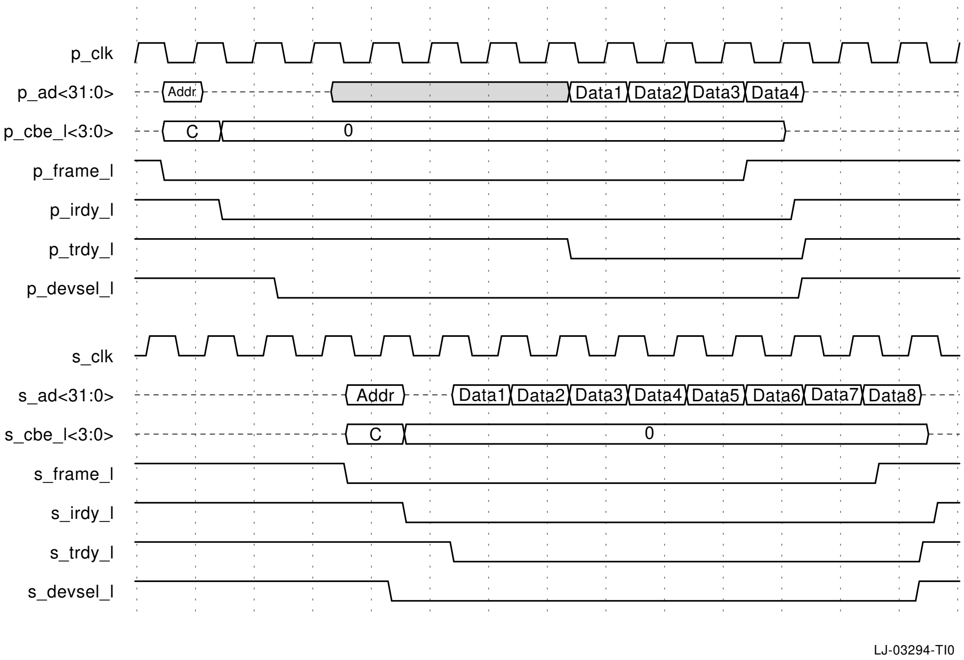


Figure 4–6 Memory Read Multiple with Extra Prefetching



4.4.4.4 The Use of Prefetchable Reads

The 21050 treats the following transactions as prefetchable:

- Memory read line
- Memory read multiple
- Memory read and address of a downstream read falls into the prefetchable address range (Section 4.6)
- Upstream read and the Prefetch Disable bit is not set in the Chip Control register in 21050 configuration space

4.4.4.5 Read Boundaries

The 21050 imposes read boundaries on prefetchable reads. When a read boundary is reached, the 21050 deasserts ***y*****frame_l** to the target at the beginning of the last data phase to indicate that only one more dword will be read. When the 21050 delivers this last longword to the initiator, the 21050 asserts ***x*****stop_l** with ***x*****trdy_l**, causing a disconnect to occur on that data phase.

Prefetchable reads are disconnected on cache line boundaries if the cache line size register is set to a value other than 0 and one of the following is true:

- Transaction is a memory read and both of these occur:
  - Read address falls into the prefetchable address range for downstream reads
  - Prefetch Disable bit is not set in the Chip Control register for upstream reads
- Transaction is a memory read line
- Transaction is a memory read multiple and both of these occur:
  - Read address falls into non-prefetchable address range for downstream reads
  - Prefetch Disable bit is set in the Chip Control register for upstream reads

Figure 4–5 shows a prefetchable read disconnecting on a cache line boundary.

Prefetchable reads are disconnected on a 256-dword boundary if:

- Transaction is a memory read and the follow is true:
  - Cache line size register has a value of 0
  - Read address falls into the prefetchable address range for downstream reads
  - Prefetch Disable bit is not set in the Chip Control register for upstream reads
- Transaction is a memory read line and the cache line size register has a value of 0.

- Transaction is a memory read multiple and
  - Cache line size register has a value of 0
  - Read address falls into the prefetchable address range for downstream reads
  - Prefetch Disable bit is not set in the Chip Control register for upstream reads

The 21050 assumes that the cache line size is a power of 2. If a value that is not a power of 2 is written into the cache line size register, the 21050 assumes the next lower power of 2 number as the cache line size (if a 6 is written, then a cache line size of 4 is used).

4.4.5 Configuration Transactions

In order to support hierarchical bridging, two types of configuration accesses are defined, Type 0 and Type 1. A Type 0 configuration transaction, designated when ***x*ad<1:0>** is set to 00, indicates that the configuration transaction is intended for a device that resides on that PCI bus. A Type 1 configuration transaction, designated when ***x*ad<1:0>** is set to 01, indicates that the configuration transaction is to be passed on to another PCI bus.

Figure 4–7 shows the address formats of Type 0 and Type 1 configuration accesses.

Figure 4–7 Configuration Address Formats

31–11	10–08	07–02	01	00
Reserved	Function Number	Register Number	0	0

Type 0

31–24	23–16	15–11	10–08	07–02	01	00
Reserved	Bus Number	Device Number	Function Number	Register Number	0	1

Type 1

LJ-03237-TI0

The 21050 can respond as the intended target to Type 0 configuration transactions initiated on the primary PCI bus. The 21050 ignores all Type 0 transactions initiated on the secondary PCI bus. Type 0 transactions are never forwarded across the 21050.

The 21050 forwards and converts Type 1 configuration transactions downstream as one of the following:

- Type 1 (on the primary PCI bus) to Type 0 (on the secondary PCI bus)
- Type 1 (on the primary PCI bus) to Type 1 (on the secondary PCI bus)
- Type 1 write (on primary PCI bus) to Special Cycle (on secondary PCI bus)

The 21050 forwards and converts Type 1 configuration transactions upstream as one of the following:

- Type 1 (on the secondary PCI bus) to Type 1 (on the primary PCI bus)
- Type 1 write (on the secondary PCI bus) to special cycle (on the primary PCI bus)

Configuration write transactions cannot be posted. Transactions are disconnected after transfer of the first dword. The initiator is held in wait states by the 21050 until data is successfully transferred to the target. Configuration read transactions do not use prefetching. Transactions are disconnected after transfer of the first dword, and no additional dwords are read from the target during the transaction.

4.4.5.1 Type 0 Configuration Transactions

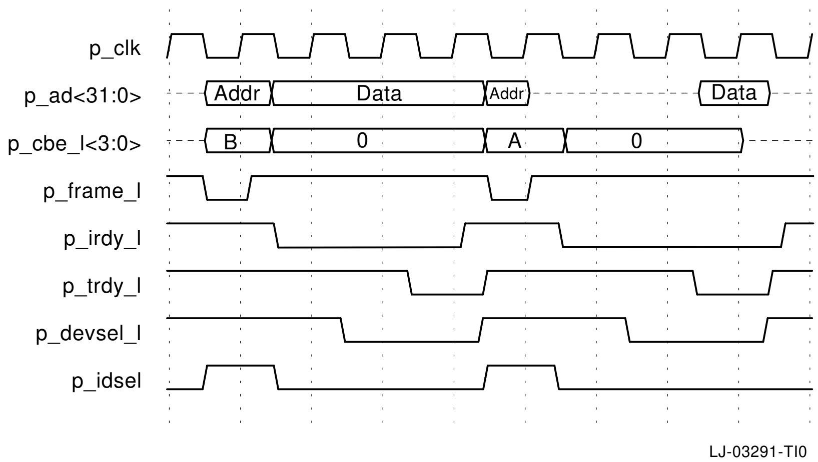
The 21050 responds only to Type 0 transactions as the intended target on the primary PCI bus. All other Type 0 transactions are ignored. The 21050 responds as the intended target to a Type 0 transaction on the primary bus when, during the address phase the following occurs:

- **p_idsel** is asserted.
- **p_ad<1:0>** is set to 0.
- Function number on **p_ad<10:8>** is set to 0.

If these conditions are not met, then the 21050 ignores the transaction. If these conditions are satisfied, the 21050 performs an access to its configuration state. The register number addresses a particular dword in 21050 configuration space.

Configuration access to the 21050 configuration space is disconnected after one dword transfer. Figure 4–8 shows a configuration read and write access to the 21050.

Figure 4–8 Type 0 Configuration Write Access and Configuration Read Access



4.4.5.2 Type 1 Configuration Transactions

In order to determine whether a Type 1 configuration transaction will be forwarded or converted, bus numbers are assigned to each PCI bus in the system. The following bus numbers are programmable in the 21050 configuration space:

- Primary bus number—designates the bus number for the 21050 primary bus
- Secondary bus number—designates the bus number for the 21050 secondary bus
- Subordinate bus number—designates the highest bus number of a PCI bus behind the 21050

The range defined by the secondary bus number (inclusive) and the subordinate bus number (inclusive) specify the range of all buses that are downstream from the 21050.

Before any configuration accesses can be made to the secondary PCI bus or to any other hierarchical PCI buses on the secondary side of the 21050, the secondary bus number of this 21050 must be initialized in the 21050 configuration space.

4.4.5.3 Type 1 to Type 0 Conversion

The 21050 forwards a Type 1 transaction downstream and converts it to a Type 0 transaction when the transaction is intended for a device on the secondary PCI bus. A Type 1 transaction is forwarded downstream and converted to a Type 0 transaction when the following conditions are met during the address phase:

- **p_ad<1:0>** is 01, indicating a Type 1 transaction.
- Value on **p_ad<23:16>** matches the secondary bus number.

If these conditions are met, the 21050 asserts **p_devsel_l** to indicate that the cycle has been accepted. The 21050 generates a Type 0 configuration transaction on the secondary bus with **s_ad<1:0>** equal to 0 and one of the **s_ad<31:16>** lines, acting as idsel signals on the secondary bus is asserted.

Table 4–3 shows the appropriate address line to be asserted is derived from the device number driven on **p_ad<15:11>**.

Table 4–3 Device Number to s_ad Signal Mapping

Device Number p_ad<15:11>	Secondary IdSel s_ad<31:16>	Device Number p_ad<15:11>	Secondary IdSel s_ad<31:16>
00000	0000 0000 0000 0001	01000	0000 0001 0000 0000
00001	0000 0000 0000 0010	01001	0000 0010 0000 0000
00010	0000 0000 0000 0100	01010	0000 0100 0000 0000
00011	0000 0000 0000 1000	01011	0000 1000 0000 0000
00100	0000 0000 0001 0000	01100	0001 0000 0000 0000
00101	0000 0000 0010 0000	01101	0010 0000 0000 0000
00110	0000 0000 0100 0000	01110	0100 0000 0000 0000
00111	0000 0000 1000 0000	01111	1000 0000 0000 0000
		10000 - 11110	0000 0000 0000 0000
		11111	Generate Spec. Cyc. (R.No. = 00h) 0000 0000 0000 0000 (R.No. > 01h)

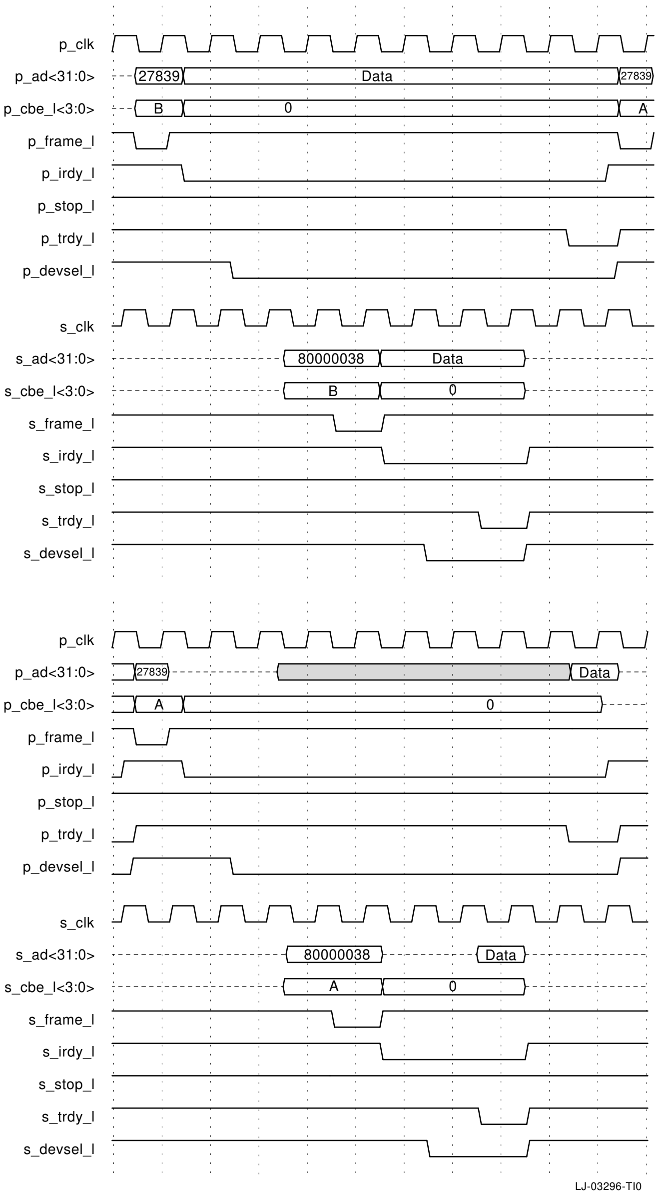
The 21050 can generate Type 0 configuration cycles for up to 16 devices. Configuration transactions with device numbers greater than 0Fh can still be forwarded, but the 21050 will not assert any of the **s_ad<31:16>** lines acting as secondary bus idsel lines. Such a transaction is likely to result in a master abort, unless some external assertion of a secondary bus idsel line is performed. One cycle of additional address stepping is used when driving a Type 0 address in order to allow time for the secondary bus idsel line to the target to become valid.

Note

The 21050 never performs a Type 1 to Type 0 conversion upstream.

Figure 4–9 shows Type 1 to Type 0 forwarding of configuration read and write transactions.

Figure 4–9 Type 1 to Type 0 Configuration Write and Read



4.4.5.4 Type 1 to Type 1 Forwarding

When the 21050 receives a Type 1 transaction that is intended for any device that is not on the secondary bus, it forwards the transaction as Type 1. Type 1 to Type 1 transactions may be forwarded upstream or downstream. In this case, the address is forwarded exactly as received on the initiating bus; no conversion is performed.

A Type 1 transaction is forwarded downstream as a Type 1 transaction when the following conditions are met during the address phase:

- The value on **p_ad<1:0>** is equal to 01b.
- The value on **p_ad<23:16>** is greater than the secondary bus number, but less than or equal to the subordinate bus number.

If these conditions are met, the 21050 asserts **p_devsel_l** to indicate that the transaction is accepted. The 21050 generates a Type 1 transaction on the secondary interface. At least two levels of PCI-to-PCI bridging are crossed before ***x*trdy_l** can be returned to the initiator for writes or reads, because the target device resides on a bus downstream from the secondary PCI bus.

A Type 1 transaction is forwarded upstream as a Type 1 transaction when the following conditions are met during the address phase:

- The value on **p_ad<1:0>** is equal to 01b.
- The value on **p_ad<23:16>** is not equal to the primary bus number, and is outside of the range defined by the secondary and subordinate bus numbers.

This indicates that the transaction is intended for a device upstream of the 21050, but not on the 21050 primary bus.

4.4.5.5 Type 1 to Special Cycle Conversion

Configuration write transactions can be used to pass special cycle messages throughout the PCI bus hierarchy. Type 1 configuration transactions can be converted and forwarded as special cycles in either direction. The write data is used as the special cycle message.

The 21050 forwards a Type 1 configuration write from the primary bus and converts it to a special cycle on the secondary bus when the following conditions are met during the address phase:

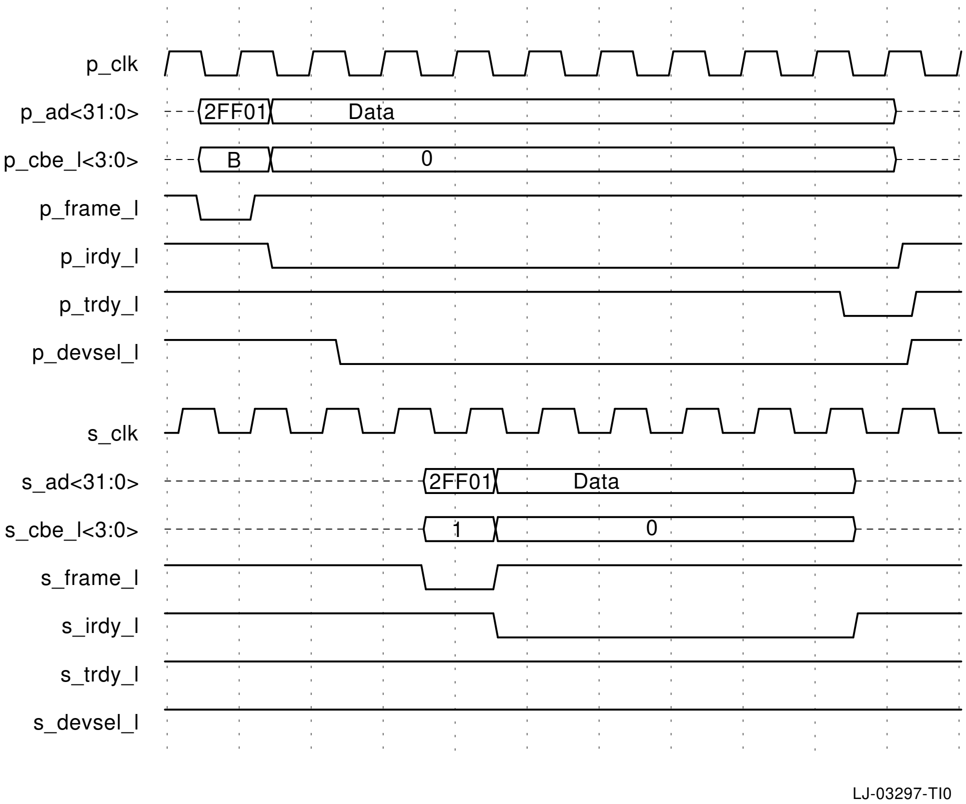
- The device number on **p_ad<15:11>** is equal to all 1's.
- The function number on **p_ad<10:8>** is equal to all 1's.
- The register number on **p_ad<7:2>** is equal to all 0's.

- **p_ad<1:0>** is equal to 01b.
- The value on **p_ad<23:16>** is equal to the 21050 secondary bus number.

If these conditions are met, the 21050 generates a special cycle on the secondary bus using the same address and data that was used for the primary configuration write.

Figure 4–10 shows the write forwarded as a special cycle downstream.

Figure 4–10 Type 1 Configuration Write to Special Cycle Downstream



The 21050 forwards a Type 1 configuration write from the secondary bus and converts it to a special cycle on the primary bus when the following conditions are met during the address phase:

- The device number on **s_ad<15:11>** is equal to all 1’s.
- The function number on **s_ad<10:8>** is equal to all 1’s.

- The register number on **s_ad<7:2>** is equal to all 0's.
- **s_ad<1:0>** is equal to 01b.
- The value on **s_ad<23:16>** is equal to the 21050 primary bus number.

If these conditions are met, the 21050 generates a special cycle on the primary bus using the same address and data used for the primary configuration write.

4.4.6 Special Cycles

The 21050 ignores all special cycle transactions generated on either the primary or secondary PCI buses.

The 21050 provides a mechanism to generate special cycles on either the primary or secondary PCI bus through a configuration write transaction (Section 4.4.5.5).

4.4.7 Interrupt Acknowledge

The interrupt acknowledge transaction is a read operation to the interrupt controller. It is assumed that the interrupt controller resides on the primary side of the 21050. Therefore, all interrupt acknowledge transactions are ignored by the 21050.

4.5 Transaction Termination

The 21050 supports both initiator-initiated terminations and target-initiated terminations as both initiator and target. A transaction is complete when both *x***frame_l** and *x***irdy_l** are deasserted.

4.5.1 Initiator Termination

There are two types of initiator termination:

- Normal termination
- Initiator abort—no target responds to transaction

4.5.1.1 DECchip 21050 as Target

If the 21050 is acting as a target during a transaction, then the only type of initiator termination possible is normal termination, in which the initiator deasserts *x***frame_l** to indicate that the next data transfer is the last transfer, and then deasserts *x***irdy_l** at the completion of the last data transfer. When the 21050 detects the deassertion of *x***frame_l**, the 21050 knows to end the transaction and deassert and tristate *x***trdy_l**, *x***stop_l** and *x***devsel_l** after the next data transfer.

4.5.1.2 DECchip 21050 as Initiator

When acting as the initiator, the 21050 can terminate a transaction normally or through a master abort.

4.5.1.2.1 Normal Termination

The 21050 uses normal termination whenever a target responds with assertion of ***y*devsel_l**. The 21050 terminates the following transactions normally:

- Non-posted write or non-prefetched read, when a single dword burst has been transferred
- Prefetched read, when a read boundary is reached or when the initiator deasserts ***x*frame_l** (in the latter case, up to four extra dwords may read from the target, which are discarded)
- Posted write, when the write buffer empties and the initiator completes the transaction on the initiator bus
- Prefetched read or a posted write, when the master latency timer expires and the initiator does not have the bus grant

4.5.1.2.2 Master Abort Termination

If the 21050 initiates a transaction and no target responds within five PCI clock cycles after ***y*frame_l** assertion, the 21050 terminates the transaction with a master abort, indicating an error condition. During a master abort termination, the 21050 deasserts ***y*frame_l**, and one PCI cycle later, deasserts ***y*irdy_l**. If ***y*frame_l** is already deasserted (only one dword was requested), ***y*irdy_l** can be deasserted five PCI clock cycles after ***y*frame_l** assertion.

The 21050 has two modes of responding to the initiator on the initiating PCI bus (where the 21050 is a target) when a master abort occurs on the target bus (where the 21050 is an initiator). The master abort response mode is set through the Master Abort Mode bit in the Bridge Control register. If the bit is not set (default mode), then the 21050 responds to a master abort as follows:

- Sets the Received Master Abort bit in the status register corresponding to the target bus.
- For a write, asserts ***x*trdy** until one cycle after ***x*frame_l** deassertion on the initiating bus, transferring write data from the initiator. Write data is discarded.
- For a read, asserts ***x*trdy** until one cycle after ***x*frame_l** deassertion on the initiating bus and returns FFFFFFFFh for all requested data dwords.

If the Master Abort Mode bit is set, then the 21050 responds to a master abort as follows:

- Sets the Received Master Abort bit in the status register corresponding to the target bus.
- For a non-posted write, returns a target abort to the initiator. The Signaled Target Abort bit is set in the status register corresponding to the initiator bus.
- For a posted write, asserts **p_serr_l** if the **p_serr_l** driver control bit in the Command Register is set, and the **p_serr_l** Disable for Master Abort on Posted Write is not set. If **p_serr_l** is asserted, sets the Signaled System Error bit in the primary status register.
- For a read, returns a target abort to the initiator. The Signaled Target Abort bit is set in the status register corresponding to the initiator bus.

A master abort during a special cycle is not considered abnormal termination, so none of the previous scenarios apply for special cycles. No event bits are set, and the 21050 performs a normal target termination in response to the configuration write on the initiator (master) bus.

4.5.2 Target Termination

The 21050 supports the following types of target termination:

Target abort	***y*stop_l** asserted and ***y*devsel_l** deasserted indicates that the target will never be able to respond to the transaction.
Target retry	***y*stop_l** and ***y*devsel_l** asserted and ***y*trdy_l** deasserted indicates that the target is not in state ready to accept data. No data is transferred during this transaction.
Target disconnect A/B	***y*stop_l**, ***y*trdy_l** and ***y*devsel_l** asserted indicates that this data transfer is to be the last data transfer on the transaction. At least one dword is transferred during the transaction.
Target disconnect C	***y*stop_l** and ***y*devsel_l** asserted and ***y*trdy_l** deasserted after previous data transfers are made indicates that the most recent data transfer was the last one of the transaction.

4.5.2.1 Target Termination Initiated by Target

If the 21050 acts as initiator and detects a target abort, retry, or disconnect through assertion of ***y*stop_l**, then the 21050 terminates the transaction on the target bus as soon as possible by deasserting ***y*frame_l** and asserting ***y*irdy_l**. The 21050 then deasserts ***y*irdy_l** one cycle later.

The 21050 responds in different ways on the initiator bus to target terminations on the target bus, depending on the type of transaction. See Section 4.5.2.1.1 for a summary of 21050 response to target terminations on the target bus.

If a target abort is detected on the target bus, the 21050 does the following:

- Sets the Received Target Abort bit in the status register corresponding to the target bus.
- If the transaction is a read or a non-posted write, returns target abort to the initiator and sets the Signaled Target Abort bit in the status register that corresponds to the initiator bus.
- If the transaction is a posted write, then the 21050:
  - If the **p_serr_l** driver enable is set and the **serr** Disable for Target Abort on Posted Write is not set, asserts **p_serr_l**
  - If **p_serr_l** is driven, sets the Signaled System Error bit in the Primary Status Register.
  - Discards any remaining write data; no attempt at delivery.

Figure 4–11 shows a target abort during an I/O write, and Figure 4–12 shows a memory read with target abort.

Figure 4–11 I/O Write with Target Abort

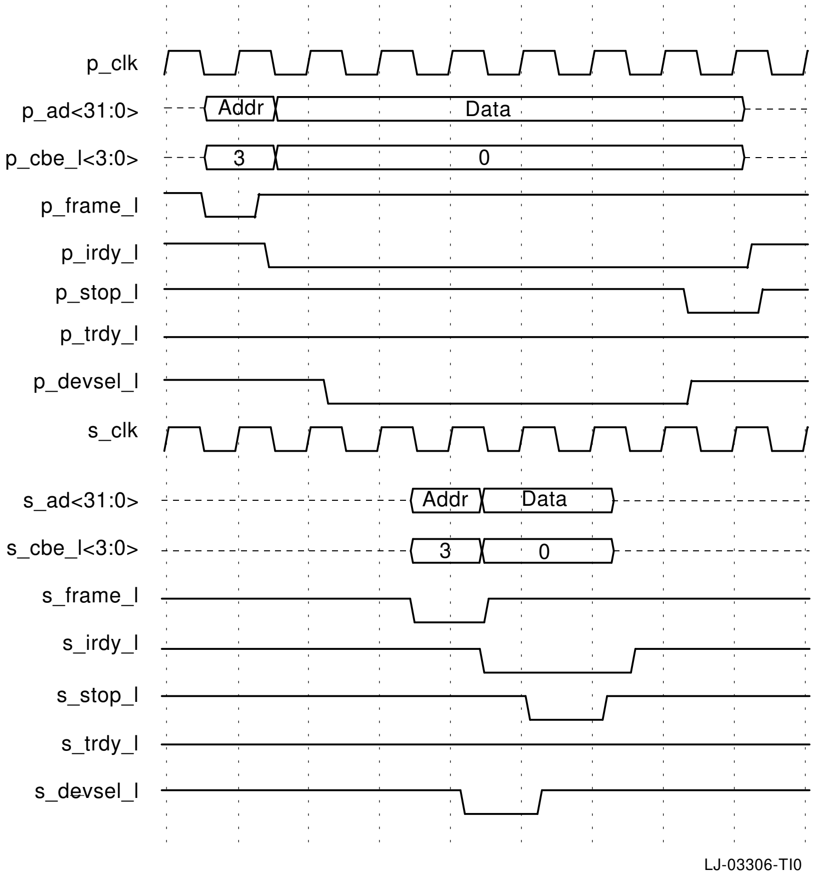
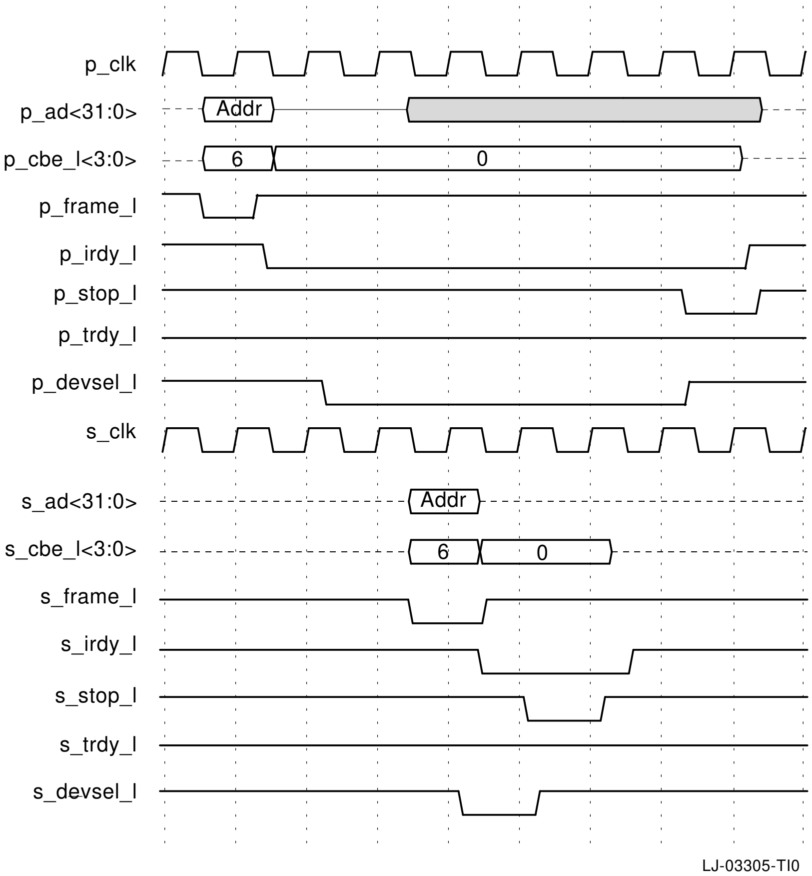


Figure 4–12 Downstream Memory Read with Target Abort



If a target retry is detected on the target bus, the 21050 does the following:

- If the transaction is a read or non-posted write, returns a target retry to the initiator. No data is transferred to or from the initiator or target.
- If the transaction is a posted write, allows up to a maximum of eight dwords to be transferred from the initiator to 21050 write buffers before returning a target disconnect.

Figure 4–13 shows a target retry during an I/O write, and Figure 4–14 shows a memory read with target retry.

Figure 4–13 Non-Posted I/O Write with Target Retry

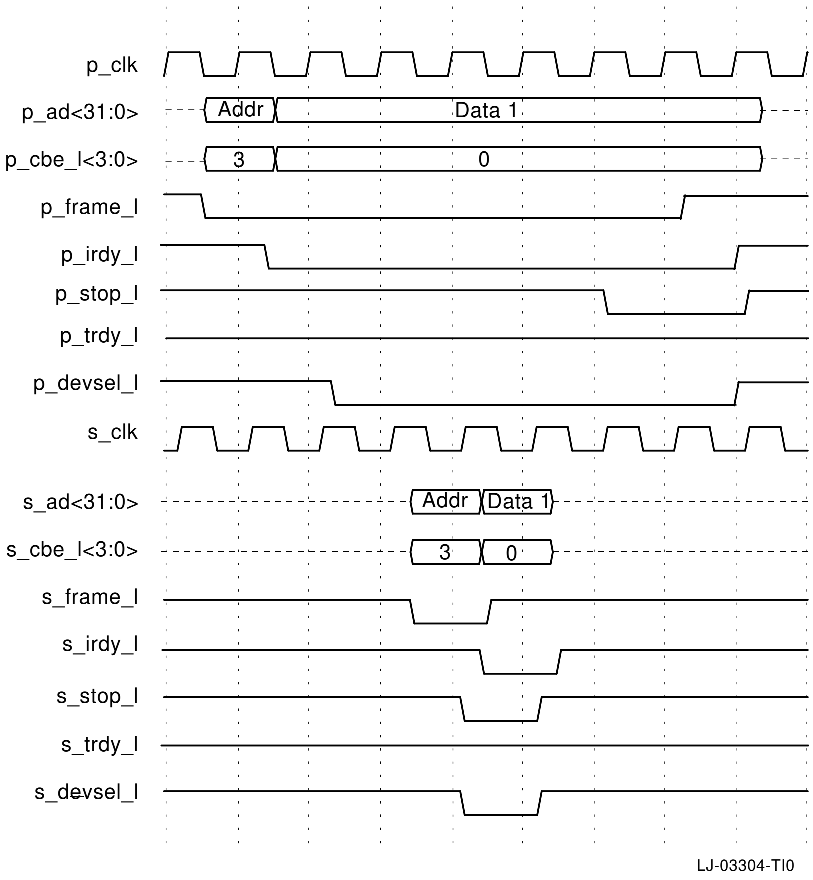
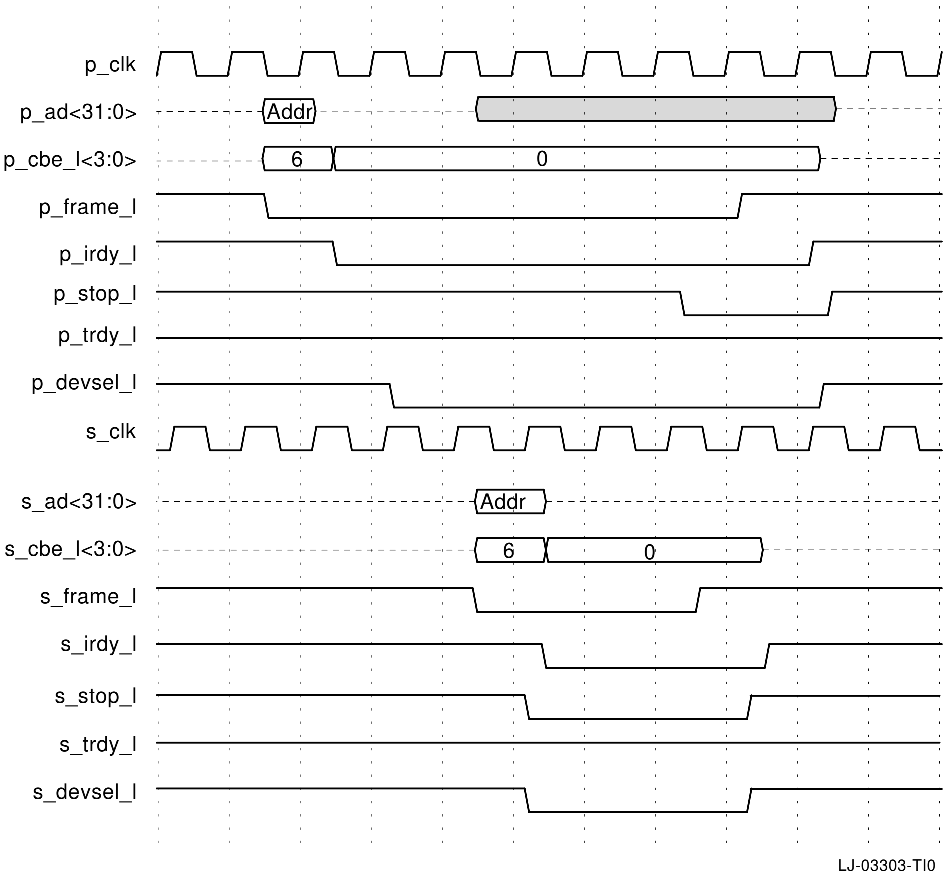


Figure 4–14 Memory Read with Target Retry



If a target disconnect is detected on the target bus, the 21050 does the following:

- If the transaction is a non-prefetched read or non-posted write takes no additional action as the 21050 restricts these types of transactions to a single dword transfer.
- If the transaction is a prefetched read, then the 21050 returns a target disconnect to the initiator after all requested data read from the target is transferred to the initiator.
- If the transaction is a posted write, then the 21050 allows up to a maximum of 8 dwords to be transferred from the initiator to 21050 write buffers before returning a target disconnect (until the write buffer fills).

4.5.2.1.1 Target Termination Summary

The following table summarizes the 21050 response to target termination scenarios for different types of transactions.

Transaction Type	Target Termination	Chip Action
Posted write	Retry	Accepts up to 8 dwords until write buffer is full. Attempts to deliver write data up to 2^{24} times; if unsuccessful, asserts **p_serr_l**.
	Disconnect	Accepts up to 8 dwords until write buffer is full. Attempts to deliver write data up to 2^{24} times; if unsuccessful, asserts **p_serr_l**.
	Abort	Returns target abort to initiator if posting not complete. Asserts **p_serr_l** if the **serr_l** disable for this condition is not set and the **p_serr_l** driver enable is set.
	No response	Maintains assertion of ***x*trdy_l** to initiator; data is not delivered. Asserts **p_serr_l** if the **serr** disable for this condition is not set and the **p_serr_l** driver enable is set.
Non-posted write	Retry	Returns target retry to initiator.
	Disconnect	Because the 21050 is limited to one burst, it always returns target disconnect to initiator after one data transfer.
	Abort	Returns target abort to initiator.
	No response	Mode 0: Asserts ***x*trdy_l** to initiator; data is not delivered. Mode 1: Returns target abort to initiator.
Prefetched read	Retry	Returns target retry to initiator.
	Disconnect	Returns target disconnect to initiator.
	Abort	Returns target abort to initiator.
	No response	Mode 0: Asserts ***x*trdy_l** to initiator; returns FFFFFFFFh. Mode 1: Returns target abort to initiator.
Non-prefetched read	Retry	Returns target retry to initiator.

Transaction Type	Target Termination	Chip Action
	Disconnect	Because the 21050 is limited to one burst, it always returns target disconnect to initiator after one data transfer.
	Abort	Returns target abort to initiator.
	No response	Mode 0: Asserts ***x*trdy_l** to initiator; returns FFFFFFFFh. Mode 1: Returns target abort to initiator.

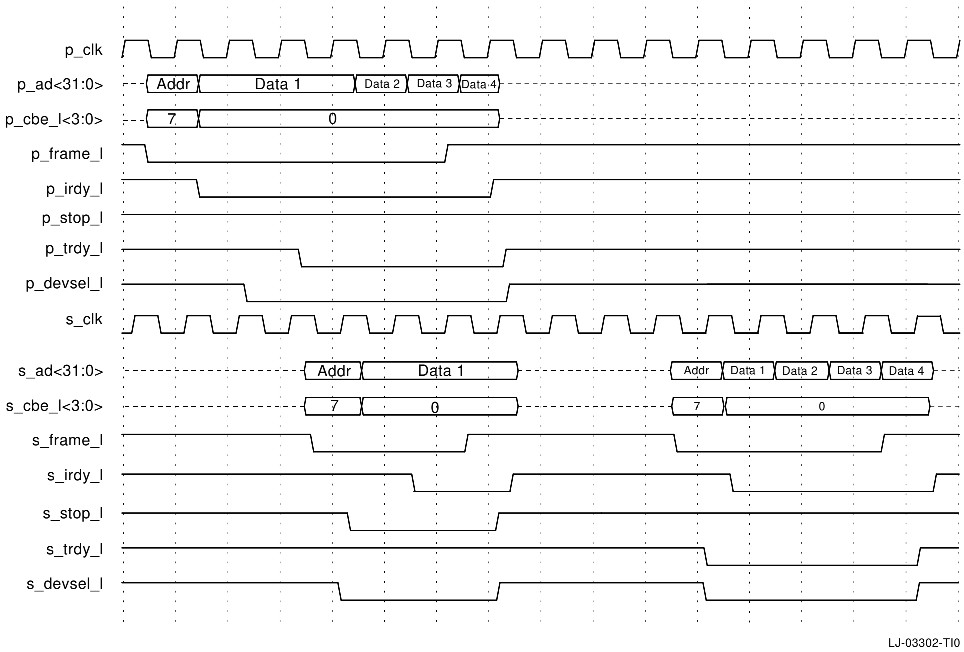
4.5.2.1.2 Delivery of Posted Write Data after Transaction Termination

If a target retry or target disconnect is received by the 21050 during a posted write, the 21050 allows the initiator to continue posting write data until the write buffer is full (a maximum of 8 dwords). After the target retry or disconnect is received by the 21050, posted write data may still remain in the write buffers that has not been transferred to the target.

When write data remains in the 21050 write buffers after a transaction to the target is terminated, whether by target termination or master latency timer expiration, the 21050 reattempts the write transaction to the target in order to empty the 21050 write buffer. The 21050 continues to attempt the write transaction until the write data is successfully transferred, or until the maximum number of attempts 2^{24} (16777216) are completed. The purpose of the write counter is to prevent indefinite hangs if the 21050 is unable to deliver write data.

Figure 4–15 shows a target retry during a posted write with subsequent write attempts.

Figure 4–15 Posted Write with Target Retry



If the maximum number of attempts are made and the 21050 has not successfully transferred the buffered data, the 21050 asserts the **p_serr_l** line, if the **p_serr_l** driver enable is set and the **serr** Disable for Delivery of Posted Write Data Failed is not set. The write data is discarded.

If the transaction is a memory write and invalidate and the write is disconnected after partial cache line transfer, remaining attempts at delivery of write data will use the memory write command because a partial cache line will be transferred.

4.5.2.2 Target Termination Initiated by 21050

The 21050 may return target retries or target disconnects to the initiator for reasons other than target retry or disconnect from the intended target.

The 21050 returns a target retry to the initiator in the following situations:

- The 21050 is involved in another transaction or there is data in one of the 21050 write buffers.

  The 21050 does allow posted writes to proceed simultaneously in opposite directions, assuming that the corresponding write buffer is empty when the transaction starts.

- The transaction is a read or non-posted write, the target wait timer is expired, and the 21050 does not have the grant for the target bus.
- The transaction is a locked transaction, but the target bus lock is already being used by another initiator.

The 21050 returns a target disconnect to the initiator for the following transactions:

- Non-posted write or non-prefetched read, and this is the first data transfer
- Posted write, and the write buffer fills (8 dwords) before the intended target returns ***y*****trdy_l**
- Posted write, and the write buffer fills (8 dwords) after expiration of the master latency timer and before the target returns ***y*****trdy_l** for a subsequent write attempt
- Posted write, and a write boundary has been reached
  - Cache line boundary for memory write and invalidate
  - 256-dword boundary for memory write
- Prefetched read, and the transaction on the target bus is terminated due to master latency timer expiration
- Prefetched read, and a read boundary is reached

  Cache line or 256-dword boundary for memory-type reads, depending on various conditions
- Burst count limit is reached

A target abort is returned to the initiator by the 21050 only when a target abort is detected on the target bus during a read or non-posted write, or an initiator abort occurs on the target bus and the Master Abort Mode bit is set.

4.6 Address Decoding

The 21050 has three programmable address ranges which determine whether memory and I/O transactions are forwarded across the 21050 or ignored. There are two address ranges for memory transactions, and a single address range for I/O transactions. The 21050 also has an ISA Compatibility mode and two modes of operation for VGA support.

There are no address holes when forwarding and filtering memory and I/O transactions. If transactions with a given address are forwarded across the 21050 from one bus, they will be ignored when initiated on the other bus.

Similarly, if transactions with a given address are ignored on one PCI bus, they will be forwarded when initiated on the other bus.

4.6.1 Memory Address Decoding

The 21050 implements two programmable memory ranges for memory transaction forwarding:

- A non-prefetchable address range specified by a Memory Base Address register (inclusive) and Memory Limit Address register (inclusive) in configuration space. Downstream memory read transactions addressing this space do not use prefetching. The space has a minimum size, granularity, and alignment of 1 megabyte.
- A prefetchable address range specified by a Prefetchable Memory Base Address register (inclusive) and Prefetchable Memory Limit Address register (inclusive) in configuration space. Downstream memory read transactions addressing this space use prefetching. The space has a minimum size, granularity, and alignment of 1 megabyte.

4.6.1.1 Memory Transaction Forwarding Based on Address Ranges

When the 21050 detects a memory read or memory write operation, it compares the address against two address ranges in order to determine whether the transaction is forwarded or not.

If the initiator is on the primary bus, then the address must fall within one of these two defined address ranges in order for the transaction to be forwarded to the secondary bus. There should be no overlap between the two ranges; if overlap exists, the guidelines for the non-prefetchable space are used.

If a memory operation is initiated on the secondary bus, the 21050 checks to see if the address falls outside of both of the address ranges described by the prefetchable and non-prefetchable memory base and limit address registers. If the address is outside of these ranges, the transaction is forwarded to the primary bus and the 21050 acts as an initiator on that bus, and as a target on the secondary bus.

4.6.1.2 Disabling a Memory Address Range

Setting the memory limit address register to a value less than its respective memory base address register turns that address range off. If both memory limit registers are set to a value less than their respective memory base address registers, the 21050 will not forward any memory transactions from the primary to secondary bus, but will forward all memory transactions initiated on the secondary bus to the primary bus. Note that the reset condition for the registers defines an address space of 00000000h—000FFFFFh in both prefetchable and non-prefetchable space for forwarding to the secondary bus.

4.6.1.3 VGA Frame Buffer Support

The 21050 also performs some memory address decoding and forwarding for frame buffer memory. If the frame buffer memory range is enabled by the VGA Enable bit in the Bridge Control register configuration space, transactions initiated on the primary bus with memory addresses in the range 000A0000h—000BFFFFh will be forwarded downstream, and transactions initiated on the secondary bus whose addresses fall within that range will be ignored. Read operations to frame buffer memory will be treated as prefetchable.

4.6.1.4 Dual Transaction Addressing

Dual address transactions are never forwarded downstream, and are always forwarded upstream. No address decoding is performed.

4.6.2 I/O Address Decoding

The 21050 implements a single programmable memory range for I/O transaction forwarding specified by an I/O Base Address register (inclusive) and I/O Limit Address register (inclusive) in configuration space. The space has a minimum size, granularity, and alignment of 4 kilobytes and a maximum I/O space of 64 kilobytes.

4.6.2.1 I/O Transaction Forwarding Based on Address Range

If an I/O transaction is initiated on the primary bus, and the address falls inside the range defined by the I/O Base Address register and I/O Limit Address register, the 21050 forwards the transaction to the secondary bus. The 21050 acts as an initiator on the secondary bus and as a target on the primary bus.

If an I/O read or write transaction is initiated on the secondary bus, the 21050 forwards the transaction to the primary bus if the address falls outside the address range described by the I/O Base Address register and I/O Limit Address register.

Any transactions that address over 64K of I/O space are ignored on the primary PCI bus and forwarded upstream from the secondary PCI bus.

4.6.2.2 Disabling the I/O Address Range

If the I/O Limit Address register is set to a value less than the I/O Base Address register, the 21050 will not forward any I/O transactions from the primary to secondary bus, but will forward all I/O transactions initiated on the secondary bus to the primary bus. Note that the reset condition for the registers defines an address space of 0000h—0FFFh for forwarding to the secondary bus.

4.6.2.3 ISA Mode

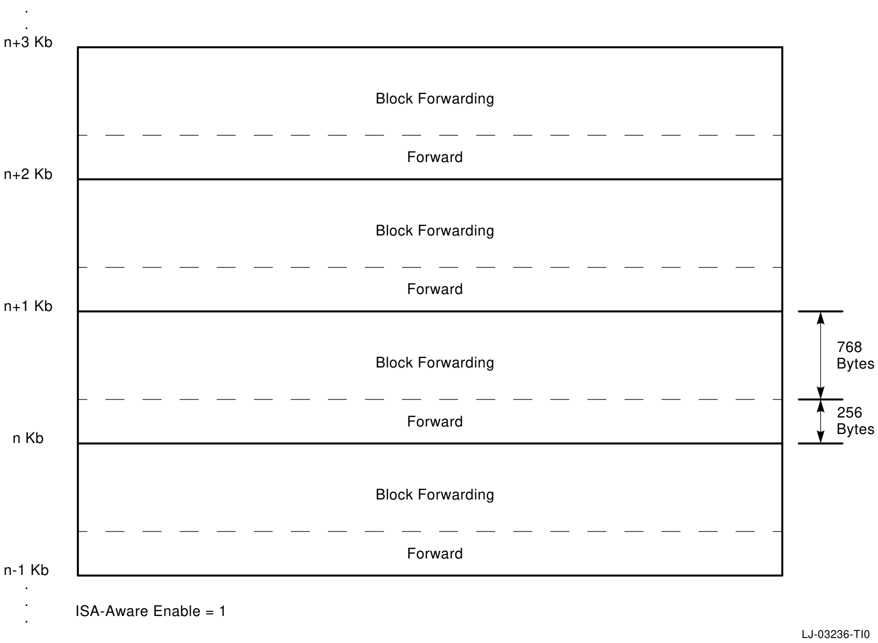
The 21050 may be programmed to be ISA-bus aware through the ISA Mode bit in the Bridge Control register in configuration space. This feature enables a finer granularity filtering mechanism for I/O read and write transactions. ISA I/O target cards decode only the lowest 10 bits of the address, which represents 1K of address space. The upper 768 bytes of this 1-kilobyte address space are allocated to general I/O ISA target cards. Because these devices do not decode the upper address bits <15:10>, their addresses are aliased and repeated 64 times, for every 1-kilobyte chunk in the 64-kilobyte address space. Thus, if an ISA card is present it would respond to its address in every 1-kilobyte chunk.

Because of address aliasing, non-ISA devices should not use I/O addresses in the upper 768 bytes of each 1-kilobyte chunk if an ISA card is present. Likewise, the 21050 should not forward transactions in this address range if an ISA card is present, regardless of whether the address falls within the I/O base and limit address range.

Only transactions addressing the first 256 bytes of each 1-kilobyte chunk can be forwarded. When ISA Mode Enable is set, the 21050 prevents the forwarding of I/O transactions that address the top 768 bytes of each 1-kilobyte chunk in I/O space. This blocking mechanism affects only those transactions initiated on the primary bus. The 21050 will not respond to blocked transactions on the initiating (primary) bus. However, as a consequence, addresses of I/O transactions initiated on the secondary bus that fall into the upper 768-kilobyte range will be forwarded upstream, regardless of where the address falls with respect to the I/O base and limit address range.

If the ISA Mode Enable bit is not set, transactions that address the upper 768 bytes of each 1-kilobyte chunk of I/O space are forwarded or filtered normally according to the I/O Base Address and I/O Limit Address registers. Figure 4–16 shows the ISA mode for downstream forwarding when the enable is set. In order to correctly use the ISA mode, the I/O Base and Limit address register should be set so that there is no overlap with ISA or EISA system-specific addresses in the lower 256 bytes of the first 4 kilobytes of I/O space.

Figure 4–16 Downstream I/O Forwarding in ISA Mode



4.6.3 VGA I/O Support

The 21050 has two modes of VGA support:

- VGA Mode for forwarding all VGA memory and I/O accesses enabled when the VGA Enable bit is set in the Bridge Control register in configuration space
- VGA Snoop Mode for *snooping* (forwarding) VGA palette I/O writes, enabled when the VGA Snoop bit is set in the Bridge Command register in configuration space

4.6.3.1 VGA Mode

When VGA Mode is enabled, the 21050 forwards downstream all memory transactions that address frame buffer memory and all I/O transactions that address VGA I/O space (Section 4.6.1.3).

In VGA Mode, any I/O transactions in the 03B0h—03DFh range, with the exception of printer port byte addresses 3BCh—3BFh, are forwarded downstream when initiated on the primary bus, and ignored if initiated on the secondary bus. When VGA Mode is enabled, bits <31:16> of the I/O address must be 0 in order for the 21050 to recognize and forward VGA I/O transactions. Bits <15:10> will not be decoded. VGA I/O addresses are aliased in every 1K block of I/O space. These aliased transactions are forwarded.

This VGA I/O forwarding is independent of the I/O address region and the ISA Mode filtering. This means that whenever the VGA bit is set, the transactions previously described are forwarded regardless of whether they fall inside the I/O Base and Limit Address range, and regardless of whether the ISA Mode bit is set.

If the VGA enable bits are not set, the 21050 uses the Base and Limit Address registers for memory and I/O forwarding, and VGA addresses are forwarded or filtered depending on whether they fall within the base and limit address ranges. If ISA Mode is enabled and VGA Mode is disabled, VGA I/O transactions are not forwarded downstream, since the addresses fall within the upper 768-byte address range.

4.6.3.2 VGA Palette Snooping

When the VGA Snoop Mode bit is set, the 21050 positively decodes and forwards downstream VGA palette register writes. VGA palette register writes use I/O byte addresses 03C6h, 03C8h, and 03C9h. Bits <31:16> must be 0 while bits <15:10> may be any value. If VGA Snoop Mode is enabled, the 21050 ignores any write transactions initiated on the secondary bus that use these three addresses. The 21050 ignores read transactions from the primary bus that use these addresses, but forwards upstream those initiated on the secondary bus.

The 21050 does not check byte enables when these I/O addresses are forwarded, but forwards byte enables as generated by the initiator. The VGA Palette Snoop Mode should not be used with VGA mode. However, if both bits are set, VGA Mode behavior is used.

4.7 Latency

Under normal data transfer conditions, there will be two PCI clock cycles of delay between data driven on one PCI bus and data driven on the opposite bus. Increased latency occurs after the address phase of a transaction, with the exception of posted writes because the 21050 must gain access to the target bus and detect a response from the target before any data transfers may proceed. This latency increases for every level of PCI-to-PCI bridging that the transaction uses. If the 21050 target wait timer is enabled and expires before the 21050 gains access to the target bus, the 21050 performs a target retry to the initiator and deasserts its request for the target bus.

For multiple burst read or write transactions, data transfers may occur on every PCI clock cycle. Data buffering in the 21050 allows it to *hide* the effects of 21050 latency on throughput. With the exception of the first data phase, the total length of all stall conditions remains the same on both the initiator and target buses. No extra stall cycles are added as a result of 21050 latency. Due to 21050 latency and prefetching, up to four extra reads from the target may be performed at the end of a prefetched read transaction.

4.7.1 Master Latency Timer

The 21050 implements two programmable master latency timers.

- The primary master latency timer—used when the 21050 acts as an initiator on the primary PCI bus.
- The secondary master latency timer—used when the 21050 acts as an initiator on the secondary PCI bus.

The appropriate master latency timer is started when the 21050 asserts ***y*frame_l**. If the master latency timer expires and the bus grant is deasserted, the initiator must deassert ***y*frame_l** at the beginning of the next data phase, with the possible exception of memory write and invalidate transactions. If the grant is still asserted when the latency timer expires, the initiator may retain control of the bus and continue the transaction until the grant deasserts.

Upon termination of the transaction on the target bus due to master latency timer expiration, the 21050:

- For a read, returns a target disconnect to the initiator when the last dword read from the target is driven on the initiating bus.
- For posted memory write, allows the initiator to continue posting data until the write buffer is full, and then returns a target disconnect. Write data is delivered to the target with subsequent memory write transactions initiated by the 21050.
- For a memory write and invalidate, performs no extra action because the initiator may terminate these transactions only on cache line boundaries. The 21050 always terminates a memory write and invalidate on the first cache line boundary.

4.7.2 Target Wait Timer

The 21050 implements two programmable target wait timers, one for the primary interface and one for the secondary interface. The appropriate timer is started when the 21050 asserts ***x*devsel_l** to the initiator. If the 21050 is waiting for access to the target bus before the transaction can proceed on the initiator bus, the target wait timer is expired, and the 21050 does not have the target bus grant, the 21050 deasserts its request for the target bus and returns a target retry on the initiator bus.

If the target wait timer expires and the 21050 has the bus grant but cannot yet start the transaction because the target bus is still busy, no action occurs and the 21050 may continue with the transaction. However, if the grant is subsequently deasserted before the 21050 can start its transaction, the 21050 must deassert its request and return a target retry to the initiator.

The maximum value for this timer is 2^8 PCI cycles, which is approximately 7.6 microseconds for a 33-megahertz system. This feature is disabled if the register contains a value of 0.

4.7.3 Burst Limit Counter

The 21050 provides a programmable burst limit counter. This allows the user to limit the number of data transfers per transaction. For example, the user may wish to use the burst limit counter to limit power dissipation for thermal reasons. Data bursts can be limited to 1 to 64 bursts per transaction. A value of 0 in this register places no burst limit on transactions. If the burst counter expires during a transaction, then the 21050 terminates the transaction normally on the target bus by deasserting ***y*frame_l**, and returns a target disconnect to the initiator.

The burst counter applies to both upstream and downstream transactions.

4.8 Deadlock Avoidance

The following rules apply to transactions that cross the 21050:

- In most cases, the current transaction crossing the 21050 must be complete on both interfaces before the next transaction to be forwarded can proceed. However, there are some exceptions:
  - Posted writes may be transferred to the 21050 on both interfaces simultaneously, although data is driven to the target on one interface at a time.
  - Configuration access to 21050 configuration space will be allowed to proceed if an upstream posted write is in progress or if there is data in upstream write buffers.

- If transactions initiated close to the same time on both the primary and secondary buses are intended for targets on the opposite interface, and at least one of the transactions is not a postable write,
  - The transaction initiated first is forwarded across the 21050, and the 21050 performs a target retry in response to the second transaction.
  - If both transactions are initiated during the same PCI cycle, the transaction initiated on the primary bus is forwarded across the 21050, and the 21050 performs a target retry in response to the secondary bus transaction.

4.8.1 Read Operations

For all read operations, if a read is already initiated on one PCI bus intended for a target on the opposite PCI bus, and a second transaction, also intended to cross the 21050, is initiated on the opposite bus before the read occurs, the 21050 issues a retry to the initiator of the second transaction.

4.8.2 Posted Write Operations

For posted write transactions, in order to prevent potential deadlock conditions in a hierarchical PCI bus environment, the 21050 allows memory write data to be posted simultaneously in both directions.

If the target bus is not idle by the time the write buffer fills, the write transaction is disconnected. This applies to both the upstream and the downstream transactions.

4.8.3 Non-Posted Write Operations

For all non-posted write operations, if a non-posted write is already initiated on one PCI bus intended for a target on the opposite PCI bus, and a second transaction, also intended to cross the 21050, is initiated on the opposite bus before the non-posted write occurs, the 21050 issues a retry to the initiator of the second transaction.

4.9 Data Synchronization

Data synchronization should be handled by the device or device driver as described in the *PCI Specification*. The 21050 cannot guarantee that write buffers are flushed before interrupts are delivered to the processor. However, the 21050 does implement two external pins, **s_bufne_l** and **s_dispst_l**, which may aid in data synchronization. These pins indicate buffer empty status and control write posting (Section 4.15).

4.10 Exclusive Access

The 21050 supports the resource lock mechanism to achieve exclusive access to a target for transactions that cross the 21050. Primary and secondary bus lock mechanisms are assumed to be concurrent, except when a locked transaction crosses the 21050.

4.10.1 Acquiring Exclusive Access

If the initiator wants to create a lock on a target on the other side of the 21050, the initiator must first gain access to the ***x*lock_l** signal on its bus. A lock is gained on the initiating bus when the initiator:

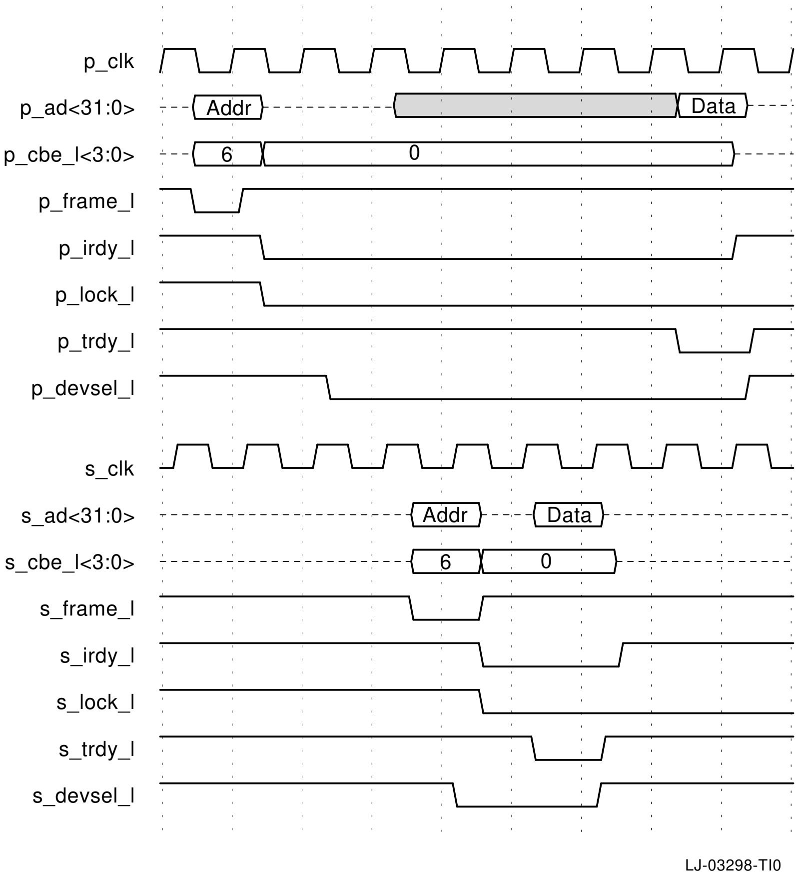
- Acquires access to the initiator bus while ***x*lock_l** is deasserted
- Deasserts ***x*lock_l** during the address phase of the transaction
- Asserts ***x*lock_l** one cycle later

After the initiator gains the lock on its bus, the 21050 attempts to gain access to ***y*lock_l** on the target bus. The 21050 must:

- Acquire access to the target bus while ***y*lock_l** is deasserted
- Deassert ***y*lock_l** during the address phase of the transaction
- Assert ***y*lock_l** one cycle later

Note that, in accordance with the *PCI Specification*, the first transaction of a locked sequence must be a read. Figure 4–17 shows a downstream read with lock acquisition.

Figure 4–17 Memory Read with Lock Acquisition



For a locked transaction involving a dual address transaction, the ***x*lock_l** signal should deassert during the first address phase, and reassert at the start of the second address phase.

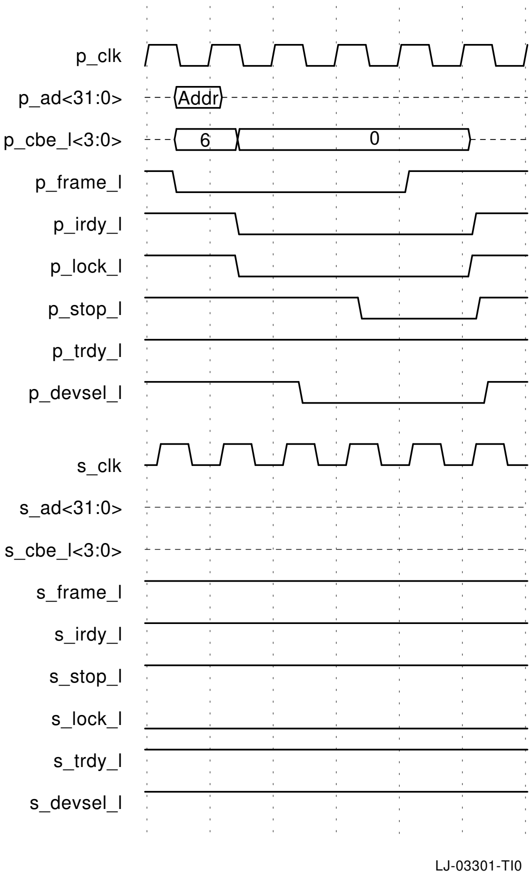
If a target retry, target abort, or master abort occurs in response to the first transaction in a locked sequence before any data is transferred, the initiator must relinquish the lock by deasserting ***x*lock_l**. This implies that, if the intended target returns a target retry or target abort, causing the 21050 to relinquish the lock on the target bus, this target termination must be passed back to the initiator in order to force the initiator to relinquish the lock on the initiator bus. Because the first transaction of a locked series of transactions is a read, the 21050 normally performs this action.

In some situations, the 21050 causes a target retry to occur in response to a lock acquisition transaction by the initiator, causing the initiator to relinquish the bus:

- The 21050 chip is already forwarding another transaction.
- The ***y*lock_l** signal on the target bus is asserted, indicating that another initiator has control of the lock on the target bus (Figure 4–18).
- The target wait timer expires before access to the target bus is gained.

Figure 4–18 shows a target retry due to the target bus lock being busy.

Figure 4–18 Read with Lock Acquisition Attempt



If the 21050 detects a master abort on the target bus, this condition is returned to the initiator bus as either normal termination or a target abort. If a target abort is returned, the initiator relinquishes the lock. If normal termination occurs, then the initiator should recognize that the FFFFFFFFh read data condition indicates a master abort; otherwise the initiator might retain possession of the lock until the locked sequence ends.

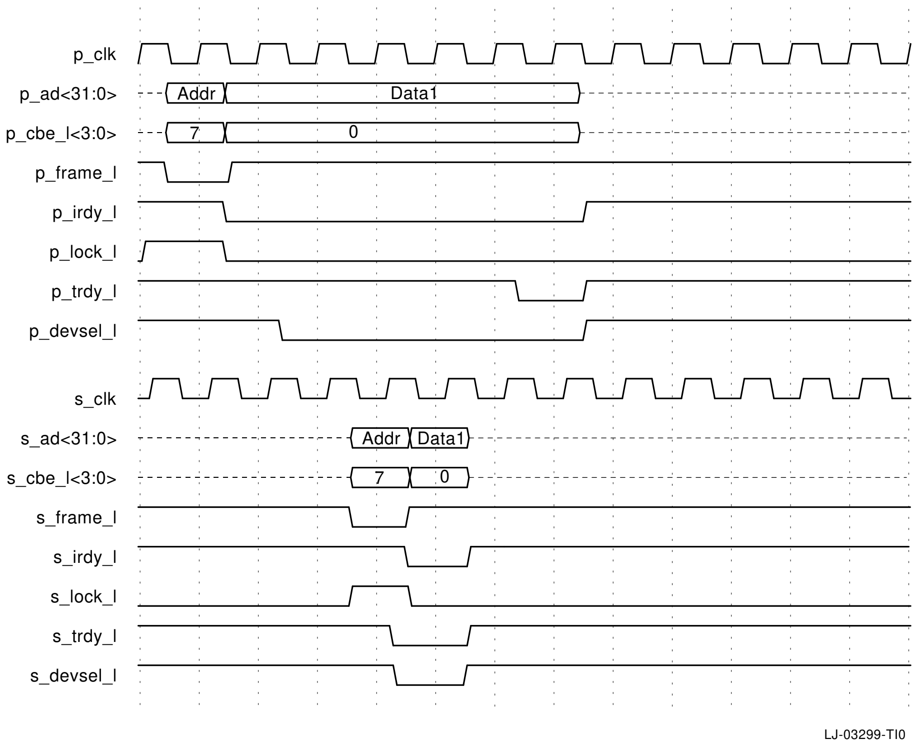
4.10.2 Maintaining Exclusive Access

After a lock is achieved on both buses, it must be maintained on both buses until the initiator relinquishes the lock or until a master abort or target abort is detected on any locked transaction. A transaction is considered a locked transaction when the initiator deasserts ***x*lock_l** at the start of the address cycle, then asserts ***x*lock_l** one phase later.

Figure 4–19 shows a downstream memory write with lock continuation.

The 21050 always relinquishes the lock on the target bus upon detection of the master abort condition. After a lock is established, if a master abort occurs on a subsequent locked write, then the 21050 relinquishes the lock on the target bus and the target becomes unlocked. However, if the 21050 is in default master abort mode, then the initiator can keep ***x*lock_l** asserted on the initiator bus because the initiator does not know the transaction on the target bus resulted in a master abort. The only effect this should have is to tie up the ***x*lock_l** signal longer than necessary. If the 21050 is configured to return target abort when a master abort is detected, both initiator and 21050 will relinquish the initiator and target bus locks.

Figure 4–19 Downstream Memory Write with Lock Continuation



Target aborts are passed back to the initiator for all transactions, as posted writes are not allowed when the target bus is locked. To prevent deadlocks when locks are in use, all writes are are not postable as long as ***y*lock_l** on the target bus is asserted. This allows the 21050

- To pass back any target aborts, so the lock can be relinquished on the initiator bus.
- To detect master aborts on the secondary bus and cause the initiator to relinquish the lock. The 21050 can be configured to return a target abort on the initiator bus when a master abort is detected. When in this mode, a master abort that is returned to the initiator as a target abort, will cause the initiator to relinquish the lock.
- To pass back any target retries to the initiator of the transaction, and allows the initiator that has locked the target to forward transactions across the 21050 to the locked target. This prevents deadlock if a different

initiator attempts to access a locked target across the 21050 when posted write and write data are stranded in write buffers.

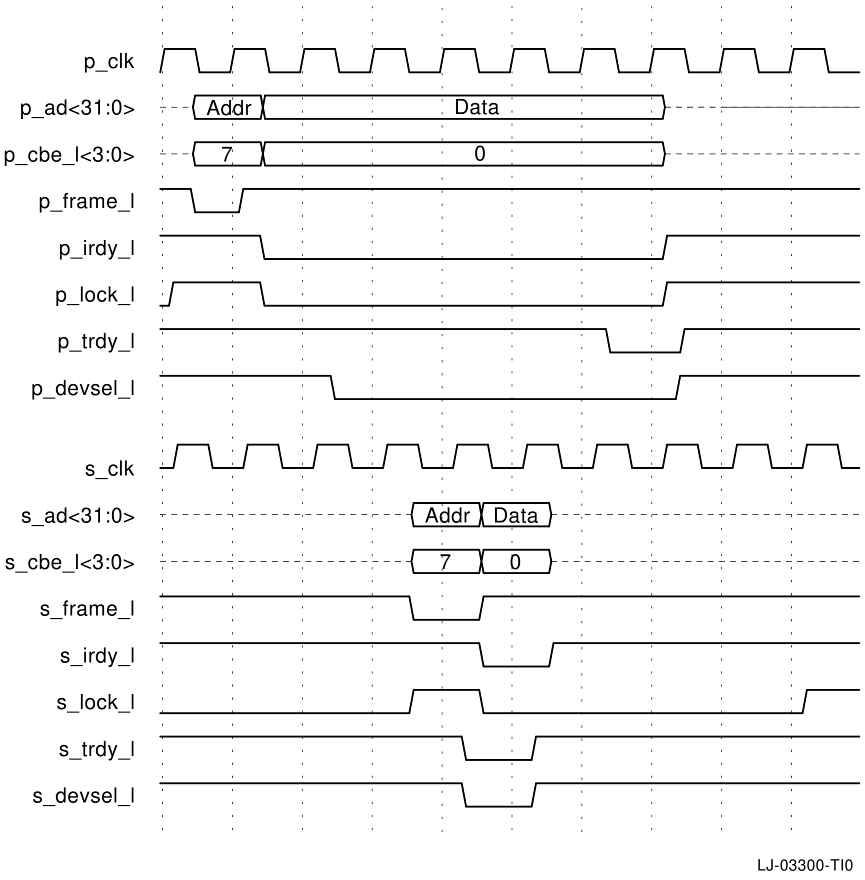
4.10.3 Ending Exclusive Access

Exclusive access is terminated when the ***x*lock_l** is deasserted at the end of a transaction. Deassertion of ***x*lock_l** must occur no earlier than the first cycle where ***x*frame_l** and ***x*irdy_l** are deasserted, ending the transaction. The ***x*lock_l** signal can be deasserted after this point, and should be deasserted as soon as possible.

When exclusive access is terminated on the last data phase of a transfer, the 21050 may not be able to release the lock on the target bus until two cycles after the deassertion of ***x*lock_l** on the initiator bus. This delay is due to the 2-clock-cycle latency of the 21050. Therefore, the ***y*lock_l** might stay asserted for a few cycles after completion on the last data phase on the target bus.

Figure 4–20 shows a memory write which ends a series of locked transactions.

Figure 4–20 Memory Write with End of Lock



DECchip 21050 as a Locked Target

The 21050 does not support lock as an intended target.

4.11 Error Handling

The 21050 implements the data parity error signals **p_perr_l** and **s_perr_l** on both interfaces, and also implements the address parity and system error signals **p_serr_l** and **s_serr_l** on both interfaces. This section describes parity checking and generation, as well as the different conditions under which **p_serr_l** is asserted.

4.11.1 Address Parity Errors

If the 21050 detects an address parity error on either interface, then the 21050:

- Does not respond as a target to the transaction.
- Sets the Detected Parity Error bit in the Status register that corresponds to the interface where the parity error was detected.
- Asserts **p_serr_l** and set the Signaled System Error bit in the Primary Status register if:
  - The **p_serr_l** Driver Enable is set.
  - The Parity Error Response bit is set in the Command register (if detected on the primary interface) or the Bridge Control register (if detected on the secondary interface).
  - For secondary bus address parity errors, the SErr Forward Enable bit is set in the Bridge Control register.

4.11.2 Data Parity Errors on Initiator Bus

If the 21050 detects a data parity error during a write transaction on the initiator bus, then the 21050:

- Sets the detected Parity Error bit in the Status register that corresponds to the initiator bus.
- Asserts ***x*perr_l** if the Parity Error Response bit is set in the Command register (if the primary bus is the initiator bus) or the Bridge Control register (if the secondary bus is the initiator bus).
- Continues with the transaction.
- If a write transaction, forwards the bad parity condition with corresponding data to the target bus so that the target device is aware of the bad parity.

4.11.3 Data Parity Errors on Target Bus

If the 21050 detects a data parity error during a read operation on the target bus, then the 21050:

- Sets the Detected Parity Error bit in the Status register that corresponds to the target bus.
- Asserts ***y*perr_l** if the Parity Error Response bit is set in the Command register (if the primary bus is the target bus) or the Bridge Control register (if the secondary bus is the target bus).
- Sets the Data Parity Reported bit in the Status register that corresponds to the target bus if the Parity Error Response bit is set.
- Returns the bad parity with the corresponding read data back to the initiator.
- Continues with the transaction normally.

If the 21050 detects ***y*perr_l** asserted during a write operation on the target bus, the 21050 sets the Data Parity Reported bit in the Status register that corresponds to the target bus if the Parity Error Response bit is set.

If the transaction is a non-posted write and a data parity error is signaled, if possible, the 21050 asserts ***x*perr_l** on the initiator bus two cycles after write data is transferred from the initiator, if the Parity Error Response bit that corresponds to the initiator bus is set. If the timing of the transaction prevents assertion of ***x*perr_l** on the initiator bus, the 21050 asserts **p_serr_l** if:

- The Parity Error Response bit corresponding to the target bus is set.
- The **p_serr_l** Driver Enable bit is set.
- The **serr** disable for posted write parity errors is not set.

If the transaction is a posted write, then the 21050 asserts **p_serr_l** and set the Signaled System Error bit in the Primary Status register if:

- The Parity Error Response bit corresponding to the target bus is set.
- The **p_serr_l** Driver Enable bit is set.
- The **serr** disable for posted write parity errors is not set.

In this case, the 21050 cannot pass back bad parity (because it is a write) or assert ***x*perr_l** during the proper cycle (because data is posted).

Table 4–4 lists parity error signaling for transactions that are forwarded.

Table 4–4 Data Parity Errors Signals for Forwarded Transactions

Transaction	Parity Error Detected on	Initiator Bus	Target Bus	Primary Bus
Posted write	Target bus	–	***y*perr_l**[3]	**p_serr_l**[1,5]
	Initiator bus	***x*perr_l**[1]	***y*perr_l**[3, 4]	–
Non-posted write	Target bus	***x*perr_l**[1,6]	***y*perr_l**[3]	**p_serr_1**[7]
	Initiator bus	***x*perr_l**[1]	***y*perr_l**[3, 4]	–
Any read	Target bus	***x*perr_l**[2, 4]	***y*perr_l**[1]	–
	Initiator bus	***x*perr_l**[2]	–	–

[1]Signal asserted by 21050.

[2]Signal asserted by initiator.

[3]Signal asserted by target.

[4]Asserted because bad parity is passed with data across the 21050.

[5]**serr** disable for this event must be clear to assert **p_serr_l**

[6]If timing allows, otherwise assert **p_serr_l**.

[7]If **x_perr_l** cannot be asserted.

4.11.4 Other Errors

The 21050 may be configured to assert **p_serr_l** for a variety of error conditions:

- A ***y*perr_l** assertion on target bus is detected during posted write transaction.
- Inability to deliver posted write data after 2^{24} attempts, as a result of target retry or disconnect.
- A target abort is detected during a posted write transaction.
- A master abort is received during a posted write transaction.

Each condition has a corresponding disable bit in the **serr** disable register, that, when set, prevents **p_serr_l** from asserting when that condition occurs.

The 21050 has a **s_serr** Forward Enable bit in the Bridge Control register which allows forwarding of **s_serr_l** to **p_serr_l**. The 21050 sets the Detected System Error bit in the secondary status register and the signaled system error bit in the primary status register. The 21050 does not assert **s_serr_l**.

The **p_serr_l** Driver Enable bit must be set in the command register for the signal to be driven for any reason. If the 21050 drives **p_serr_l**, it also sets the Signaled System Error bit in the Primary Device Status register.

When the **p_serr_l** signal is detected as asserted, the condition that caused the assertion may be decoded by checking the conditions listed in Table 4–5.

Table 4–5 p_serr_l Assertion Conditions

p_serr_l Assertion Condition	Asserted Status Bits
s_serr_l detected as asserted	Signaled System Error in primary status register and Detected System Error in secondary status register
***x*perr_l** detected on target bus during posted write	Signaled System Error in primary status register and Detected Parity Error in either primary or secondary status register
Unable to deliver posted write data after $\mathbf{2}^{24}$ attempts	Signaled System Error in primary status register and no other **serr**-related status bits asserted in the 21050 or in secondary bus devices
Target abort detected during posted write	Signaled System Error in primary status register and Received Target Abort in either primary or secondary status register
No target response during posted write	Signaled System Error in primary status register and Received Master Abort in either primary or secondary status register

4.12 PCI Bus Arbitration

This section describes the primary and secondary PCI bus arbitrations.

4.12.1 Primary PCI Bus Arbitration

The primary bus arbitration interface is a single request output and a grant input. The 21050 does not implement any centralized arbitration functions for the primary bus. The 21050 initiates transactions on the primary bus on behalf of initiators on the secondary bus.

The 21050 asserts **p_req_l** when it needs to forward a transaction upstream. During the cycle immediately after the 21050 detects an asserted level on **p_gnt_l** and the primary bus is idle, the 21050 starts the transaction on the primary bus by

– Asserting **p_frame_l**

– Driving the address on **p_ad<31:0>**

– Driving the command on **p_cbe_l<3:0>**

– Deasserting **p_req_l**

The 21050 may deassert **p_req_l** at any time before starting a transaction, for example, due to expiration of the target wait timer.

The 21050 performs bus parking duties (driving **p_ad**, **p_cbe_l**, and **p_par** when the bus is idle) only when **p_gnt_l** is asserted and **p_req_l** is deasserted.

4.12.2 Secondary PCI Bus Arbitration

The secondary bus central function pin, **s_cfn_l**, determines whether the 21050 performs arbitration for access to the secondary PCI bus. If asserted low, the 21050 performs arbitration. Otherwise, arbitration is assumed to be performed externally.

The secondary PCI arbitration function can support up to six bus masters, in addition to the 21050.

The 21050 includes itself in the arbitration for those transactions in which it is acting on behalf of an initiator on the primary bus. Because the primary and secondary buses operate concurrently, grants are given out independently of the status of the primary bus, and independently of the primary bus arbitration scheme. However, for all transactions that cross the 21050 except posted writes, both the primary and secondary buses must be acquired before data transfers on either bus can proceed.

4.12.2.1 Secondary Bus Arbiter Protocol

The arbiter re-assigns priorities every time **s_frame_l** is asserted (every time a new transaction starts). This starts a new arbitration cycle. From this point until the next transaction starts, the arbiter asserts the grant signal that corresponds to the highest priority request that is asserted.

If a grant for a particular request is asserted and a higher priority request subsequently asserts, then the arbiter deasserts the asserted grant signal and asserts the grant that corresponds to the new higher priority request on the next PCI clock cycle.

If the secondary PCI bus is busy (either **s_frame_l** or **s_irdy_l** is asserted), then the arbiter may deassert one grant and assert another grant during the same PCI clock cycle. If the secondary PCI bus is idle, then the arbiter never asserts a grant signal in the same PCI cycle that it deasserts another. If it deasserts one grant, it asserts the next grant no earlier than one PCI clock cycle later.

If the 21050 detects that an initiator has failed to assert **s_frame_l** after 16 cycles of both grant assertion and an idle bus condition, the arbiter deasserts the grant. That initiator does not receive any more grants until it deasserts its request for at least one PCI clock cycle.

4.12.2.2 Priority Schemes

The 21050 offers two priority schemes, which can be selected by using the Arbitration Mode Select bit in the Chip Control register.

Alternating Mode

The default priority scheme selected upon reset gives the 21050 highest priority on alternate arbitration cycles. This mode is selected when the Arbitration Mode Select is low. For cycles during which the 21050 does not have highest priority, the priority rotates evenly among the six external request inputs.

Rotating Mode

If the Arbitration Mode Select bit is set, a simple rotating priority scheme is used, in which highest priority rotates evenly between the seven request inputs (six external requests and the 21050 request).

4.12.2.3 Request Mask Timer

The 21050 implements a programmable request mask timer. If the 21050 issues a target retry in response to a secondary bus initiator, the request mask timer is started. That secondary bus initiator request is then masked for the number of PCI cycles programmed in the request mask timer which is contained in the Chip Control register in 21050 configuration space. When the timer expires, that initiator’s bus request may again be serviced. If a target retry is subsequently returned by the 21050 to another secondary bus initiator before expiration of the request mask timer, the latter’s bus request is masked only for the number of PCI cycles remaining in the timer.

4.12.2.4 Secondary Bus Arbiter Disabled

If **s_cfn_l** is tied high, it is assumed that secondary bus arbitration is performed externally to the 21050 and the secondary bus arbiter is disabled. The 21050 then reconfigures the following:

- **s_gnt_l<0>** as secondary bus request output
- **s_req_l<0>** as secondary bus grant input
- **s_gnt_l<5:1>** outputs deasserted
- **s_req_l<5:1>** inputs ignored

  In this case, the 21050 secondary arbitration interface protocol is similar to the primary arbitration interface protocol.

4.12.2.5 Secondary Bus Parking

If **s_cfn_l** is asserted low, the 21050 by default drives **s_ad**, **s_cbe_l** and **s_par** when the secondary bus is idle. If **s_cfn_l** is deasserted, the 21050 only drives these signals if **s_req_l<0>**, configured as the 21050 external grant, is asserted.

Note

The **s_cfn_l** signal should be tied high or low through a resistive device since the input is connected to the diagnostic Nand tree and must be toggled during Nand tree testing.

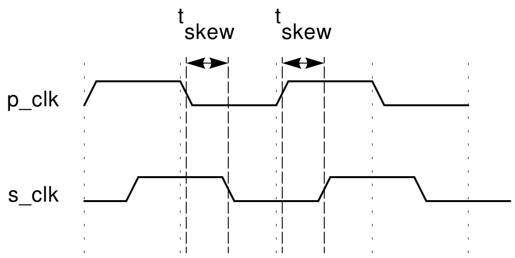
4.13 Clocks

The 21050 has two clock inputs. The **p_clk** signal is referenced to the primary interface. All primary PCI bus signals are driven and sampled with **p_clk**. The **s_clk** signal is used by the secondary interface. All secondary PCI bus signals are driven and sampled with **s_clk**.

- Both clocks operate at the same frequency and are synchronous to each other.
- The maximum clock skew between **p_clk** and **s_clk** is no greater than 7 nanoseconds.
- The minimum clock skew between **p_clk** and **s_clk** is no less than 0 nanoseconds.
- The maximum operating frequency of the clocks is 33 Megahertz.

Figure 4–21 shows the relative timing.

Figure 4–21 p_clk and s_clk Relative Timing



The 21050 provides seven outputs, **s_clk_o<6:0>**, which can be used as secondary clock outputs.

- One of these outputs can be fed back into the **s_clk** input of the 21050.
- The other six clock outputs can be used as clock inputs for secondary bus devices. If these clock outputs are used for clock distribution, the **s_clk** input must still meet the timing requirements listed in Chapter 7.

Here are suggested guidelines for using the secondary clock outputs:

- Secondary clocks should not be used to drive option card clocks or as a clock input to another 21050 chip.
- Secondary clock delay external to the 21050 should not exceed 2 nanoseconds.

4.14 Reset

This section describes the primary, secondary, and chip reset bits.

4.14.1 Primary Reset

The 21050 has one reset input, **p_rst_l**. When asserted, **p_rst_l** causes the 21050 to immediately tristate all primary and secondary PCI interface signals. The 21050 resets all registers to 0 and clears all write buffers, unless designated otherwise (Chapter 5). The **p_rst_l** asserting and deasserting edges may be asynchronous to **p_clk** and **s_clk**.

4.14.2 Secondary Reset

The 21050 is responsible for driving the secondary reset signal. There are three conditions during which **s_rst_l** can be asserted:

- When **p_rst_l** is asserted.

  The **s_rst_l** remains asserted as long as **p_rst_l** is asserted.

- When the Secondary Reset bit in the Diagnostic Control register is set.

  The **s_rst_l** signal remains asserted until the Secondary Reset bit is cleared by a configuration write.

- When the Chip Reset bit in the Diagnostic Control register is set, **s_rst_l** remains asserted until the Secondary Reset bit is cleared by a configuration write.

When **s_rst_l** is asserted, **s_ad**, **s_cbe**, and **s_par** are immediately tristated and then driven low within a couple of clock cycles, while all other secondary interface signals are tristated. The 21050 still responds to accesses to its configuration space on the primary interface. Write buffers in both directions are cleared.

4.14.3 Chip Reset

The 21050 can be reset by setting the Chip Reset bit in the Diagnostic Control register. As soon as the chip reset is completed, the Chip Reset bit is automatically cleared within six PCI clock cycles and the chip is ready for configuration. While the chip is in reset, the 21050 is inaccessible for any type of transaction to or across it.

4.15 Buffer Control

The 21050 implements one input, **s_dispst_l**, and one output, **s_bufne_l**, on the secondary interface that controls write buffer posting.

When an asserted value of **s_dispst_l** is detected, the 21050 asserts **s_bufne_l** two cycles later if write data exists in either upstream or downstream data buffers. As long as write data that was in the write buffers at the time of **s_dispst_l** assertion remains in the write buffers, **s_bufne_l** remains asserted. **s_bufne_l** does not reflect status of data that was posted subsequent to **s_dispst_l** assertion. After the data that was in the write buffers at the time of the **s_dispst_l** assertion is flushed, the 21050 deasserts **s_bufne_l**.

The 21050 can be configured to disable posting by setting the Disable Posting bit in the Chip Control register after **s_dispst_l** is asserted. In this case, posting remains disabled until **s_dispst_l** is deasserted. While posting is disabled, all writes must wait for target response before transferring data from the initiator, and are disconnected after one data transfer.

If the 21050 is configured so that posting is not disabled, the 21050 continues to accept posted writes in the normal manner, even during assertion of **s_dispst_l**. If no external control of bridge write posting is necessary, **s_dispst_l** should be tied high.

5

Configuration Register Description

This chapter provides programmer reference material for all 21050 configuration space registers. Configuration space registers include the predefined PCI device and 21050 registers, as well as other implementation-specific registers used during initialization. All registers located in configuration space are accessible only from the primary bus interface of the 21050. Figure 5–1 shows a configuration space map.

Figure 5–1 Configuration Space Map

31	16	15	00	
Device ID		Vendor ID		00h
Primary Interface Status		Primary Interface Command		04h
Class Code			Revision ID	08h
Reserved	Header Type	Primary Latency Timer	Cache Line Size	0Ch
Reserved				10h
Reserved				14h
Secondary Latency Timer	Subordinate Bus Number	Secondary Bus Number	Primary Bus Number	18h
Secondary Interface Status		I/O Limit Address	I/O Base Address	1Ch
Memory Limit Address		Memory Base Address		20h
Prefetchable Memory Limit Address		Prefetchable Memory Base Address		24h
Reserved				28h
Reserved				2Ch
Reserved				30h
Reserved				34h
Reserved				38h
Bridge Control		Interrupt Pin	Reserved	3Ch
SErr Event Disable	Burst Limit Counter	Diagnostic Control	Chip Control	40h
Reserved		Secondary Target Wait Timer	Primary Target Wait Timer	44h
Secondary Write Attempt Counter				48h
Primary Write Attempt Counter				4Ch
Reserved				50h - FFh

LJ-03284-TI0

5.1 Predefined Header Space Register Description

This section provides a detailed description of predefined configuration space registers in the 21050.

5.1.1 Device ID and Vendor ID Register

This section describes the device ID and vendor ID register.

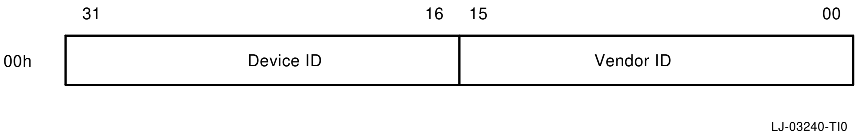


Table 5–1 Device ID and Vendor ID Register

Address	00 hex	–
Access	<31:0>	Read only.
Field description	<31:16>	Device ID identifies this device as the 21050 and is internally hardwired to be 0001h.
	<15:0>	The Vendor ID identifies Digital Equipment Corporation as the vendor of this device and is internally hard wired to be 1011h.

5.1.2 Primary Status and Primary Command Register

This section describes the primary status and primary command register.

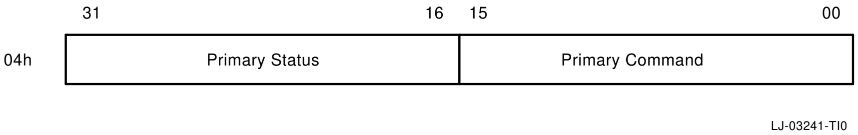


Table 5–2 Primary Status and Primary Command Register

Address	04 hex	–
Access	<31:27>	Read/write-1-to-clear.
	<26:25>	Read only.
	<24>	Read/write-1-to-clear.

(continued on next page)

Table 5–2 (Cont.) Primary Status and Primary Command Register

	<23>	Read only.
	<22:9>	Read only as 0.
	<8:5>	Read/write.
	<4:3>	Read only.
	<2:0>	Read/write.
Field Description	<31>	Detected Parity Error on primary bus.
	<30>	Signaled System Error on primary bus.
	<29>	Received Master Abort on primary bus.
	<28>	Received Target Abort on primary bus.
	<27>	Signaled Target Abort on primary bus.
	<26:25>	**p_devsel_l** timing for the 21050 acting as target on primary bus. Set to 01b to designate medium timing.
	<24>	Data Parity Detected. Set when the 21050 is acting as a master on the primary bus, **p_perr_l** is detected as asserted, or a parity error is detected on the primary bus, and the Parity Error Response bit in the primary command register is set.
	<23>	Fast back-to-back capable. Reads only as a 1 to indicate that the 21050 can respond to fast back-to-back transactions on the primary bus.
	<22:10>	Reserved. Reads only as 0.
	<9>	Fast back-to-back control. Reads only as 0 to indicate that the 21050 will not perform fast back-to-back accesses to targets on the primary bus.
	<8>	**p_serr_l** driver enable. If high, the 21050 can drive **p_serr_l**. If low, the **p_serr_l** driver is disabled. Reset to 0.
	<7>	Wait Cycle Control. Reads only as 0 to indicate that the 21050 does not do address stepping, with the exception of downstream Type 1 to Type 0 configuration transactions.
	<6>	Parity error response. When this bit is low, parity checking is disabled on the primary bus interface. Reset to 0.

(continued on next page)

Table 5–2 (Cont.) Primary Status and Primary Command Register

	<5>	VGA Snoop Enable. If set, enables VGA snooping for the 21050. The 21050 positively decodes VGA palette register I/O writes and ignores palette register I/O reads.
	<4:3>	Reserved. Will read as 0.
	<2>	Master enable. When set high, 21050 can act as a master on the primary bus. Reset to 0.
	<1>	Memory transaction target enable. When set high, the 21050 can respond as a target to memory transactions on the primary bus. Reset to 0.
	<0>	I/O Transaction target enable. When set high, the 21050 can respond as a target to I/O transactions on the primary bus. Reset to 0.

5.1.3 Class Code/Programming Interface/Revision ID Register

This section describes the class code/programming interface/revision ID register.

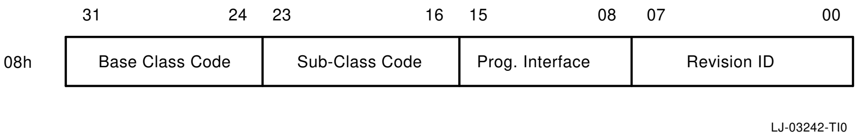


Table 5–3 Class Code/Programming Interface/Revision ID Register

Address	08 hex	–
Access	<31:0>	Read only.
Field description	<31:24>	Base class code of the device. Reads as 06h to indicate that it is a 21050 device.
	<23:16>	Sub-class code of the device. Reads as 04 hex to indicate that device is a 21050.
	<15:8>	Programming interface of the device. Reads as all 0’s.
	<7:0>	Revision ID for the 21050.

5.1.4 Primary Master Latency Timer/Cache Line Size/Header Type

This section describes the primary master latency timer (MLT)/cache line size/header type register.

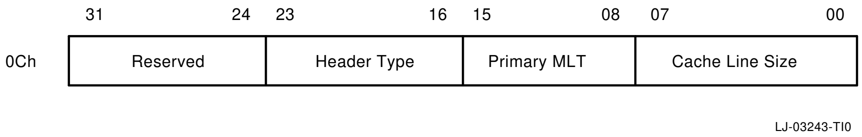
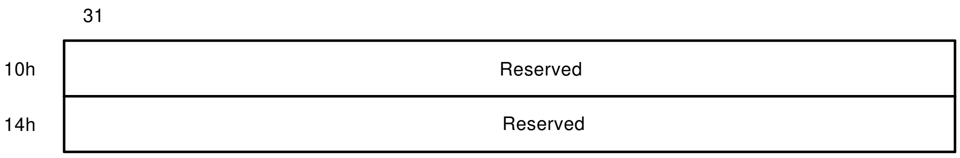


Table 5–4 Primary Master Latency Timer/Cache Line Size/Header Type Register

Address	0C hex	–
Access	<31:24>	Reads only as 0's.
	<23:16>	Read only.
	<15:11>	Read/write.
	<10:8>	Read only as 0's.
	<7:0>	Read/write.
Field Description	<31:24>	Reserved. Reads as all 0's.
	<23:16>	Header type. Defines the layout of addresses 10 hex through 3F hex in configuration space. Reads only as 01 hex to indicate that the register layout conforms to the PCI-PCI bridge layout.
	<15:8>	Master latency timer for primary interface. Designates the maximum number of PCI clock cycles from the assertion of **p_frame_l** until the expiration of the timer. The lower 3 bits are read only, giving a granularity of eight PCI clock cycles. If 0, the timer is not used. Reset to 0.
	<7:0>	Cache line size. Designates the cache line size for the system in units of 32-bit words. The cache line size is restricted to be a power of 2. The most significant 1 in the cache line size register is used to set the cache line size. Any other 1's are ignored. Used when terminating memory write and invalidate transactions and when prefetching during memory read type transactions. Reset to 0.

5.1.5 Reserved Registers (10—14 Hex)

This section describes the reserved registers (10—14 hex).



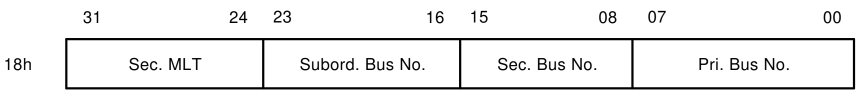
LJ-03244-TI0

Table 5–5 Reserved Registers (10—14 Hex)

Address	10—14 hex	–
Access	–	Reads only as 0’s.
Field description	–	All registers are reserved and read as 0’s.

5.1.6 Primary Bus Number/Secondary Bus Number/Subordinate Bus Number/Secondary Master Latency Timer Register

This section describes the primary bus number/secondary bus number /subordinate bus number/secondary master latency timer register.



LJ-03245-TI0

Table 5–6 Primary Bus Number/Secondary Bus Number/ Subordinate Bus Number/Secondary Master Latency Timer

Address	18 hex	–
Access	<31:27>	Read/write.
	<26:24>	Read only as 0.
	<23:0>	Read/write.
Field description	<31:24>	Master Latency Timer for secondary interface. Designates the maximum number of PCI clock cycles from the assertion of **s_frame_l** until the expiration of the timer. The lower 3 bits are read only, giving a granularity of eight PCI clock cycles. If 0, the timer is not used. Reset to 0.
	<23:16>	The subordinate bus number. Defines the inclusive upper limit of a range of bus numbers. This range determines whether configuration accesses are to be forwarded across the 21050 to the secondary bus. Reset to 0.
	<15:8>	The secondary bus number. Enables accesses to configuration space of a hierarchical PCI bus. If matched to a number driven during the Type 1 address cycle on the primary bus, the access is for a device on the secondary bus. Also provides the non-inclusive lower limit for the bus number range described above. Reset to 0.
	<7:0>	Primary bus number. Designates the primary bus. Used to decode Type 1 configuration cycles initiated on the secondary bus intended for special cycle generation on the primary bus. Reset to 0.

5.1.7 I/O Base Address/I/O Limit Address/Secondary Status Register

This section describes the I/O base address/I/O limit address/secondary status register.

	31 — 16	15 — 08	07 — 00
1Ch	Secondary Status	I/O Lim. Address	I/O Base Address

LJ-03247-TI0

Table 5–7 I/O Base Address/I/O Limit Address/Secondary Status Registers

Address	1C hex	–
Access	<31:27>	Read/write-1-to-clear.
	<26:25>	Read only.
	<24>	Read/write-1-to-clear.
	<23>	Read only.
	<22:16>	Read only as 0.
	<15:12>	Read/write.
	<11:8>	Read only as 0.
	<7:4>	Read/write.
	<3:0>	Read only as 0.
Field description	<31>	Detected Parity Error on secondary bus.
	<30>	Detected System Error on secondary bus – **s_serr_l** detected asserted.
	<29>	Received Master Abort on secondary bus.
	<28>	Received Target Abort on secondary bus.
	<27>	Signaled Target Abort on secondary bus.
	<26:25>	**s_devsel_l** timing for the 21050 acting as target on secondary bus. Set to 01b to designate medium timing.
	<24>	Data Parity Detected. Set when the 21050 is acting as a master on the secondary bus, and **s_perr_l** is detected as asserted, or a parity error is detected on secondary bus, and the Secondary Parity Response bit is set in the 21050 control register.

(continued on next page)

Table 5–7 (Cont.) I/O Base Address/I/O Limit Address/Secondary Status Registers

<23>	Fast back-to-back capable. Reads only as 1 to indicate that the 21050 can respond to fast back-to-back transactions on the secondary bus.
<22:16>	Reserved.
<15:8>	Contains bits <15:8> of the I/O limit address, or upper limit (inclusive), which defines an address range used to determine whether to forward I/O transactions from the primary to the secondary bus. Since the minimum I/O space is 4 kilobytes, the least-significant 4 bits are read only as 0h. The 16 most-significant bits of the address are assumed to be 0. The 12 least-significant bits of the address are assumed to be FFF hex. The writable bits are reset to 0.
<7:0>	Contains bits <15:8> of the I/O base address, or lower address limit, which defines an address range used to determine whether to forward I/O transactions from one PCI bus to the other. Because the minimum I/O space is 4 kilobytes, the least-significant 4 bits are read only. Both the 16 most-significant address bits and 8 least-significant address bits are assumed to be 0. This register is reset to 0.

5.1.8 Memory Base Address/Memory Limit Address Register

This section describes the memory base address/memory limit address register.

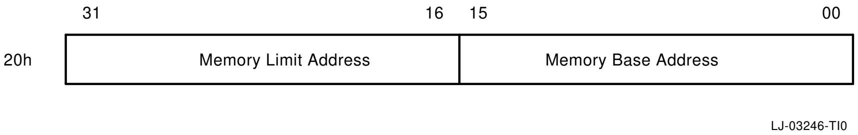


Table 5–8 Memory Base Address/Memory Limit Address Register

Address	20 hex	–
Access	<31:20>	Read/write.
	<19:16>	Read only as 0.
	<15:4>	Read/write.
	<3:0>	Read only as 0.
Field description	<31:16>	Defines the most-significant 12 bits of the upper limit (inclusive) of the memory address range. The minimum memory address range is 1 megabyte. The least-significant 4 bits are reserved and read only as 0. The 20 least-significant bits of the memory limit address should be assumed to be FFFFFh. For memory read transactions, the 21050 does not prefetch data in this address range, and memory read transactions are limited to one data burst. The writable bits are reset to 0.
	<15:0>	Indicates the upper 12 bits of the base address, or lower address limit, of a memory address range. This range is used to determine whether to forward memory accesses across the 21050. The minimum memory address range is 1 megabyte. The least-significant 4 bits are reserved and read only as 0. The 20 least-significant bits of the memory base address should be assumed to be 00000h. For memory read transactions, the 21050 does not prefetch data in this address range, and memory read transactions are limited to one data burst. This register is reset to 0.

5.1.9 Prefetchable Memory Base Address/Prefetchable Memory Limit Address Register

This section describes the prefetchable memory base address/prefetchable memory limit address register.

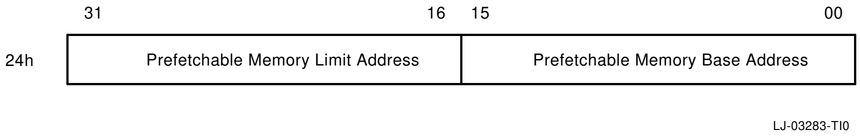
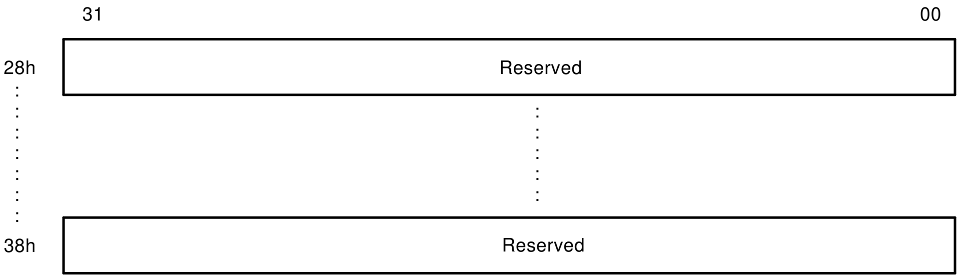


Table 5–9 Prefetchable Memory Base Address/Prefetchable Memory Limit Address Register

Address	24 hex	–
Access	<31:20>	Read/write.
	<19:16>	Read only as 0.
	<15:4>	Read/write.
	<3:0>	Read only as 0.
Field description	<31:16>	Define the most-significant 12 bits of the upper limit (inclusive) of the memory address range. The minimum memory address range is 1 megabyte. The least-significant 4 bits are reserved and read only as 0. The 20 least-significant bits of the memory limit address should be assumed to be FFFFFh. The 21050 prefetches data in this address range. The writable bits are reset to 0.
	<15:0>	Indicates the upper 12 bits of the base address, or lower address limit, of a memory address range. This range is used to determine whether to forward memory accesses across the 21050. The minimum memory address range is 1 megabyte. The least-significant 4 bits are reserved and read only as 0. The 20 least significant bits of the memory base address should be assumed to be 00000h. The 21050 prefetches data in this address range. This register is reset to 0.

5.1.10 Reserved Registers (28—38 Hex)

This section describes the reserved registers (28—38 hex).



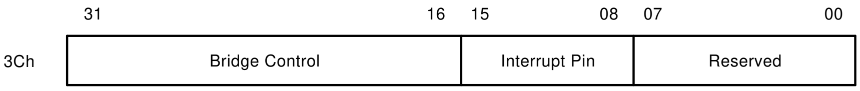
LJ-03249-TI0

Table 5–10 Reserved Registers (28—38 Hex)

Address	28h—38h	–
Access	–	Read only as 0.
Field description	–	These registers are reserved and read only as 0’s.

5.1.11 Interrupt Pin/Bridge Control Register

This section describes the interrupt pin/bridge control register.



LJ-03250-TI0

Table 5–11 Interrupt Pin/Bridge Control Register

Address	3Ch	–
Access	<31:24>	Reserved.
	<23:21>	Read/write.
	<20>	Read only as 0.
	<19:16>	Read/write.
	<15:0>	Read only.
Field description	<31:26>	Reserved. Read only as 0.
	<23>	Fast back-to-back control. Reads only as zero to indicate that the 21050 will not perform fast back-to-back accesses to targets on the secondary bus.
	<22>	Secondary Bus Reset. When set high, the 21050 tristates secondary bus outputs and immediately terminates any transactions involving the secondary interface. The **s_rst_l** signal is asserted until this bit is cleared by a configuration write. The primary bus interface is still active and the 21050 responds to accesses to its internal configuration state. The configuration state is not cleared. All write buffers will be cleared.
	<21>	Master Abort mode. When set, the 21050 signals a target abort to the initiator when a master abort is detected for reads and non-posted writes. If 0, reads return FFFFFFFFh, and write data is transferred to the 21050 and discarded. Reset to 0.
	<20>	Reserved. Read only as 0.
	<19>	VGA Enable. When set, allows the 21050 to forward transactions downstream directed at a VGA device (to memory addresses A0000h through BFFFFh and to I/O addresses 03B0h through 03DFh except 3BCh-3BFh). These addresses are independent of the memory and I/O ranges described by the memory base and address registers. Reset to 0.
	<18>	ISA Enable for I/O transactions. When set, makes the 21050 ISA-aware. When set, the 21050 blocks any forwarding of transactions addressing the last 768 bytes in each 1-kilobyte chunk of I/O address space. Applicable to I/O space within the I/O base and limit address registers only. Reset to 0.

(continued on next page)

Table 5–11 (Cont.) Interrupt Pin/Bridge Control Register

<17>	**s_serr_l** forward enable. If high, allows forwarding of **s_serr_l** on secondary address parity errors to the primary bus via assertion of **p_serr_l**. Reset to 0.
<16>	Parity Error Response. When low, disables parity checking on the secondary bus interface. Reset to 0.
<15:8>	Interrupt pin. Designates which interrupt signal the device uses. Reads only as a 00h, because the 21050 does not generate an interrupt.
<7:0>	Reserved. Read only as 0.

5.2 Implementation-Specific 21050 Register Descriptions

This section offers a detailed description of implementation-specific configuration registers.

5.2.1 Chip Control/Diagnostic Control/Burst Limit Counter/serr Disable Register

This section describes the chip control/diagnostic control/burst limit counter /serr disable register.

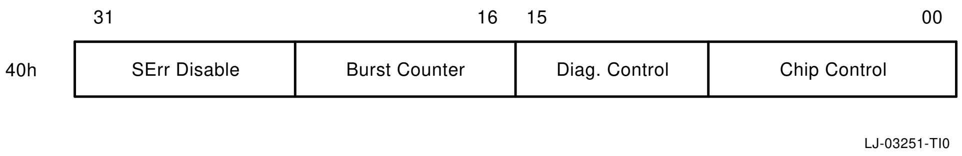


Table 5–12 Chip Control/Diagnostic Control/Burst Limit Counter/serr Disable Register

Address	40h	–
Access	<31:0>	Read/write.
Field description	<31:24>	**serr** disable. When set, the following bits prevent **p_perr_l** from being asserted for the following conditions:

Bits	Description
<31:29>	Reserved
<28>	Master abort on posted write
<27>	Target abort during posted write
<26>	Unable to deliver posted write data
<25>	Bad data parity on target bus for posted write
<24>	Reserved

	<23:16>	Burst Limit Counter. Specifies the maximum number of data transfers that can occur during any transaction. The top 2 bits are hard-wired to be 0, leaving a maximum burst range from 1 to 64 data transfers. If 0, no burst limits are applied. Reset to 0.
	<15:13>	Reserved. Read only as 0.

(continued on next page)

Table 5–12 (Cont.) Chip Control/Diagnostic Control/Burst Limit Counter/serr Disable Register

<12>	**p_serr_l** diagnostic mode assertion control. When set to 1, the 21050 asserts the **p_serr_l** signal in response to all address phases on the primary PCI bus. This bit should be used for diagnostic purposes only. Reset to 0.
<11>	**s_perr_l** diagnostic mode assertion control. When set to 1, the 21050 asserts the **s_perr_l** signal in response to all write data phases on the secondary bus intended for the 21050 or a primary bus device. This bit should be used for diagnostic purposes only. Reset to 0.
<10>	**p_perr_l** diagnostic mode assertion control. When set to 1, the 21050 asserts the **p_perr_l** signal in response to all write data phases on the primary bus intended for the 21050 or a secondary bus device. This bit should be used for diagnostic purposes only. Reset to 0.
<9>	Test mode. Used for chip test. When set to 1, all counters greater than 4 bits will be sub-divided into parallel 4 bit chunks by forced carry signals. Reset to 0.
<8>	Chip Reset. When set, the 21050 performs a chip reset. All configuration registers are returned to their reset state and the 21050 must be reconfigured. This bit is cleared by the chip upon completion of reset. **s_rst_l** is asserted until the secondary reset bit is cleared.
<7:5>	Reserved. Reads as 0.
<4>	Secondary bus prefetch disable. If not set, memory read transactions initiated on the secondary bus use prefetching. If set, memory read transactions are disconnected after one data transfer. Memory read lines and memory read multiple transactions always use prefetching and may consist of multiple data transfers. Reset to 0.
<3:2>	Request mask timer. Designates the maximum value of the request mask timer, which is enabled after the 21050 issues a target retry to a master on the secondary bus. Reset to 0. 00b—mask timer not used 01b—16 PCI clock cycles 10b—32 PCI clock cycles 11b—64 PCI clock cycles

(continued on next page)

Table 5–12 (Cont.) Chip Control/Diagnostic Control/Burst Limit Counter/serr Disable Register

<1>	Disable Posting Enable. If set, posting is disabled while **s_dispst_l** is asserted. Reset to 0.
<0>	Secondary PCI bus arbitration mode select. When set, selects rotating priority between all secondary bus requests. If not set, arbitration is also rotating priority, except that the 21050 has highest priority on alternate arbitration cycles. Reset to 0.

5.2.2 Primary Target Wait Timer/Secondary Target Wait Timer Register

This section describes the primary target wait timer/secondary target wait timer register.

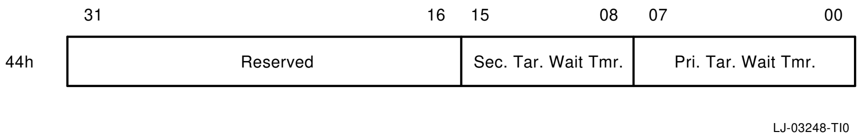


Table 5–13 Primary Target Wait Timer/Secondary Target Wait Timer Register

Address	44h	–
Access	<31:16>	Read only as 0’s.
	<15:0>	Read/write.
Field descriptions	<31:16>	Reserved. Read only as 0’s.
	<15:8>	Secondary interface target wait timer. A programmable register which indicates the maximum number of PCI cycles that the 21050 waits to gain access to the target (primary) bus for a transaction initiated on the secondary bus. If the timer expires before the 21050 receives a grant for the primary bus, the 21050 issues a target retry to the initiator on the secondary bus and deasserts its primary bus request. If 0, the 21050 waits indefinitely. Reset to 0.
	<7:0>	Primary interface target wait timer. A programmable register which indicates the maximum number of PCI cycles that the 21050 waits to gain access to the target (secondary) bus for a transaction initiated on the primary bus. If the timer expires before the 21050 receives a grant for the secondary bus, the 21050 issues a target retry to the initiator on the primary bus and deasserts its secondary bus request. If 0, the 21050 waits indefinitely. Reset to 0.

5.2.3 Secondary Write Attempt Counter Register

This section describes the secondary write attempt counter register.

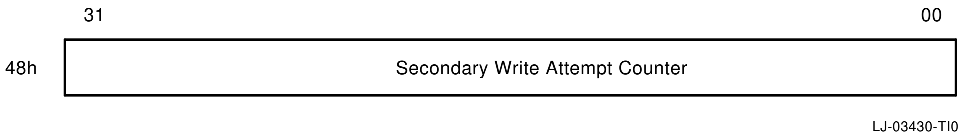


Table 5–14 Secondary Write Attempt Counter Register

Address	48h	–
Access	–	Read only
Field description	–	A read-only register which shows the number of attempts that have been made to empty data in the downstream write buffer. Used for diagnostic purposes. Reset to 0.

5.2.4 Primary Write Attempt Counter Register

This section describes the primary write attempt counter register.

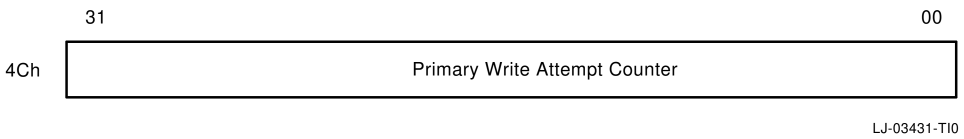


Table 5–15 Primary Write Attempt Counter Register

Address	4Ch	–
Access	–	Read only
Field description	–	A read-only register which shows the number of attempts that have been made to empty data in the upstream write buffer. Used for diagnostic purposes. Reset to 0.

5.2.5 Reserved Registers (50h—FFh)

This section describes the reserved registers (50h—FFh).

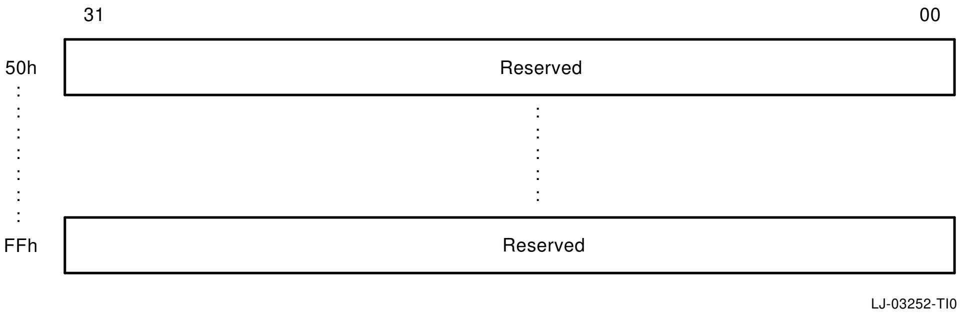


Table 5–16 Reserved Registers (50h - FFh)

Address	50h - FFh	–
Access	–	Read only as 0.
Field description	–	Reserved and read only as 0’s.

6 Diagnostics and Test Mechanisms

This chapter provides testing and diagnostics information.

6.1 Test Pins and Nand Tree Implementation

The 21050 implements two pins for testing purposes.

- The **goz_l** pin, when asserted, tristates all bidirectional pins.
- The **nand_out** pin is the output of a serial Nand tree connecting all chip inputs, except **p_clk** and **s_clk**. A pattern can be applied to chip inputs, and the **nand_out** pin verifies input pin interconnect.

When using the Nand tree test mechanism, all bidirectional signals must first be tristated by assertion of **goz_l**. The **goz_l** signal should remain asserted for the duration of this test.

Note

Any inputs tied high or low, **s_cfn_l** for example, should be connected to power or ground through a resistive device to allow use of the Nand tree feature.

The Nand tree begins at the **s_dispst_l** input and runs clockwise to **p_rst_l**, and then is output at **nand_out**.

The suggested Nand tree test sequence is:

1. Drive **goz_l** low.
2. Drive each input and bidirectional pin high, with the possible exception of **p_clk** and **s_clk**. (**p_clk** and **s_clk** are not included in the Nand tree).
3. Starting with pin 90 (**s_dispst_l**), individually drive each pin low; **nand_out** should toggle with each pin. Proceed clockwise, driving each subsequent pin low.
4. Turn off tester drivers.

5. Drive **goz_l** high.
6. Reset chip before proceeding with further testing.

6.2 Diagnostic Control Register

The 21050 implements three bits for ***x*perr_l** and **p_serr_l** control in the Diagnostic Control register, which exists in 21050 configuration space.

Caution

These bits should never be set during normal chip operation or initialization.

When the ***x*perr_l** control bit is set, the 21050 returns ***x*perr_l** to the master of any write transaction occurring on that bus. The 21050 asserts **p_perr_l** in response to all write transaction data phases initiated on the primary bus and directed to or across the 21050 when the **p_perr_l** control bit is set. Similarly, when the **s_perr_l** control bit is set, the 21050 asserts **s_perr_l** in response to all write transaction data phases initiated on the secondary bus and directed across the 21050 chip.

When the **p_serr_l** control bit is set, the 21050 returns **p_serr_l** to the master in response to any address phase occurring on the primary bus. The 21050 continues to respond to transactions, unless a real address parity error is detected.

The 21050 also implements a test mode bit in the Diagnostic Control register to facilitate chip fault test. When set, all counters in excess of 8 bits are broken up into 4-bit chunks by a forced carry signal. The 4-bit chunks of any particular counter will count in parallel when the chip is in test mode. The test mode bit should not be set unless all 21050 state machines are idle, that is, there are no transactions in progress across the 21050 chip.

7

DECchip 21050 Specifications

This chapter describes the mechanical and electrical specifications of the 21050.

7.1 Mechanical Specifications

The following sections describe the mechanical specifications of the 21050.

7.1.1 DECchip 21050 Package

The DECchip 21050 PCI chip is packaged in a 208-pin PQFP. Table 7–1 lists the mechanical specifications, and Figure 7–1 shows the package dimensions of the DECchip 21050.

Table 7–1 Lead Counts and Dimensional Attributes

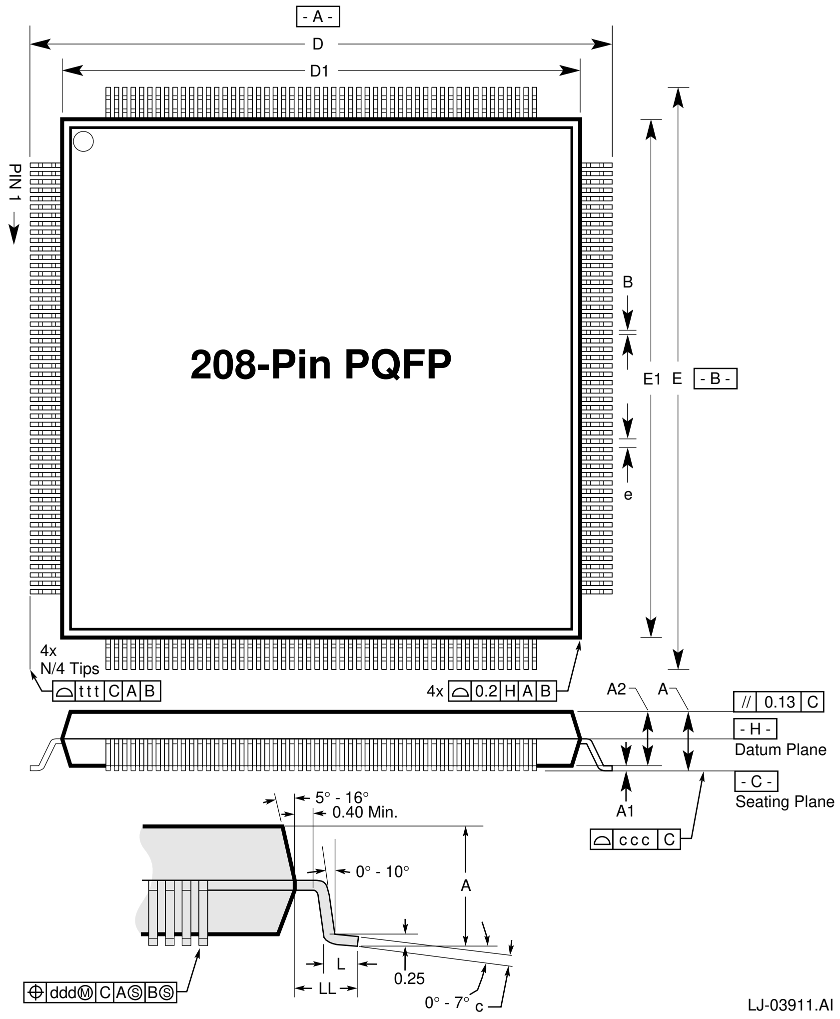
Symbol	Limit	Dimensions in Millimeters
LL	REF	1.3
e	BSC	0.5
L	MIN	0.5
L	MAX	0.75
A	MAX	4.2
A1	MIN	0.25
A2	MIN	3.17
A2	MAX	4.25
b	MIN	0.17
b	MAX	0.27
c	MIN	0.09

(continued on next page)

Table 7–1 (Cont.) Lead Counts and Dimensional Attributes

Symbol	Limit	Dimensions in Millimeters
c	MAX	0.2
ccc	–	0.08
ddd	–	0.08
ttt	–	0.2
D	BSC	30.6
D1	BSC	28
E	BSC	30.6
E1	BSC	28
R	MIN	0.08
R	MAX	0.25

Figure 7–1 Package Dimensions



7.1.2 Absolute Maximum Ratings

This section lists the absolute maximum ratings.

V_{DD}	5.0 V ±5%
T_J	100°C
T_A	40°C
P_{WC}	1.7W
Storage temperature	–55°C to 125°C

7.2 Electrical Specifications

The following sections describe the electrical specifications of the 21050.

7.2.1 Interface Signal DC Electrical Specifications

Table 7–2 defines the dc parameters met by all 21050 signals.

Table 7–2 DC Specifications

Symbol	Parameter	Condition	Minimum	Maximum	Units
V_{ih}	Input high voltage	–	2.0	V_{DD} + 0.5 V	V
V_{il}	Input low voltage	–	–0.5	0.8	V
I_{ih}	Input high leakage current[1]	V_{in} = 2.7 V	–	70	µA
I_{il}	Input low leakage current[1]	V_{in} = 0.5 V	–	–70	µA
V_{oh}	Output high voltage	I_{out} = –2 mA	2.4	–	V
V_{ol}	Output low voltage[2]	I_{out} = 3 mA, 6 mA	–	0.55	V
C_{in}	Input pin capacitance	–	–	10	pF
C_{IDSEL}	**p_idsel** pin capacitance	–	–	8	pF
C_{clk}	**p_clk**, **s_clk** pin capacitance	–	5	12	pF

Footnotes:

1. Input high leakage current and input low leakage current include I_{ozl} or I_{ozh} leakage current for bidirectional signals.

2. Most output low voltage signals have 3 mA current. The following output low voltage signals have 6 mA low output current:

p_frame_l	**p_trdy_l**	**p_irdy_l**	**p_devsel_l**
p_stop_l	**p_serr_l**	**p_perr_l**	**p_lock_l**
s_frame_l	**s_trdy_l**	**s_irdy**	**s_devsel_l**
s_stop_l	**s_perr_l**	**s_lock_l**	

7.3 Interface Signal AC Electrical Specifications

Table 7–3 Shared Signal Output Parameters

Symbol	Parameter	Condition	Min.	Max.	Units
I_{oh} (AC)	Switching current high[1]	$0 < V_{out} \leq 1.4$	-44	–	µA
		$1.4 < V_{out} \leq 2.4$	[2]	[3]	µA
	Test point	$V_{out} = 3.1$ V	–	-142	µA
I_{ol} (AC)	Switching current low[1]	$V_{out} \geq 2.2$	95	–	µA
		$2.2 > V_{out} > 0.55$	V_{out} / 0.023	[4]	µA
	Test point	$V_{out} = 0.71$ V	–	206	µA
I_{cl}	Low clamp current	$-5 < V_{in} \leq -1$	[5]	–	µA
t_r	Unloaded output rise time	0.4 V to 2.4 V	1	–	V/ns
t_f	Unloaded output fall time	0.4 V to 2.4 V	1	–	V/ns

Footnotes:

1. See V/I curves in the *PCI Specification*.
2. $-44 + (V_{out} - 1.4)/0.024$

3. $11.9 * (V_{out} - 5.25) * (V_{out} + 2.45)$ *for* $V_{DD} > V_{out} > 3.1$ V
4. $78.5 * (V_{out} * (4.4 - V_{out})$ *for* $0 \text{ V} < V_{out} < 0.71$ V
5. $-25 + (V_{in} + 1) / 0.015$

7.3.1 Interface Signal AC Timing Specifications

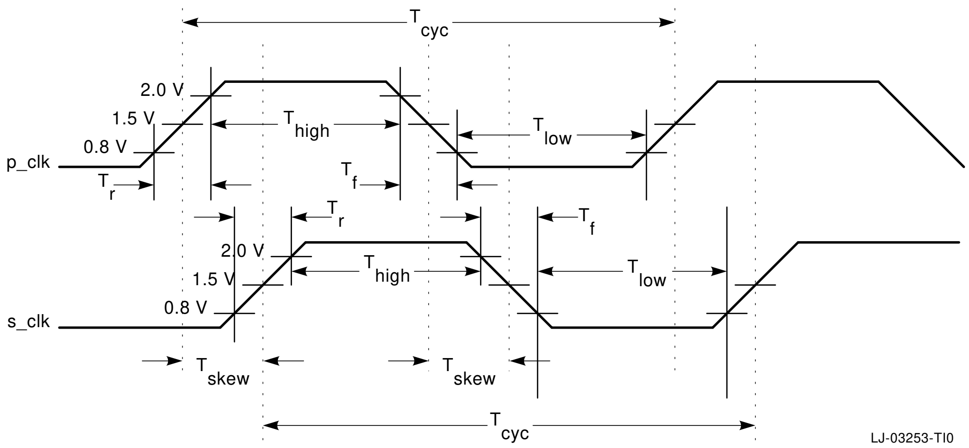
Table 7–4 and Figure 7–2 show the **p_clk** and **s_clk** ac timing.

Table 7–4 p_clk and s_clk AC Timing

Symbol	Parameter	Minimum	Maximum	Units	Notes
T_{cyc}	***x*clk** cycle time	30	–	ns	–
T_{high}	***x*clk** high time	12	–	ns	@1.5 V and @1 V/ns
T_{low}	***x*clk** low time	12	–	ns	@1.5 V and @1 V/ns
	Slew rate	1	4	V/ns	0.4 to 2.4 V
T_{skew}	Delay from **p_clk** to **s_clk**	0	7	ns	@ 1.5 V
T_{sclkr}	**p_clk** rising to **s_clk_o<6:0>** rising	0	5	ns	@1.5 V[1]
T_{sclkf}	**p_clk** falling to **s_clk_o<6:0>** falling	0	5	ns	@1.5 V[1]

[1]Measured with 30 pF lumped load.

Figure 7–2 p_clk and s_clk AC Timing



7.3.2 Input Signal AC Timing Specifications

Table 7–5 and Figure 7–3 show the input signal ac timings.

Table 7–5 Input Signal AC Timings

Symbol	Parameter	Minimum	Maximum	Units
T_{val}	***x*clk**-to- ***x*signal** valid delay—bused signals[1], [2], [3]	2	11	ns
T_{val} (ptp)	***x*clk**- to-***x*signal** valid delay—point-to-point[1], [2], [3]	2	12	ns
T_{on}	Float-to-active delay[1]	2	–	ns
T_{off}	Active-to-float delay[1]	–	28	ns

[1]All primary interface signals are used by **p_clk** and all secondary interface signals are used by **s_clk**.

[2]Minimum times measured with 0-pF equivalent load. Maximum times measured with 50-pF equivalent load.

[3]Point-to-point signals are **p_req_l**, **s_req_l<7:0>**, **p_gnt_l**, **s_gnt_l<7:0>**, **s_dispst_l**, **s_bufne_l**, and **s_cfn_l**.
All other PCI signals are shared.
All ***x*gnt_l** signals, **s_dispst_l**, and **s_cfn_l** have a setup time of 10 ns.
***x*req_l** has a setup time of 12 ns.

(continued on next page)

Table 7–5 (Cont.) Input Signal AC Timings

Symbol	Parameter	Minimum	Maximum	Units
T_{su}	Input setup time to ***x*clk**— bused signals[1], [3]	7	–	ns
T_{su} (ptp)	Input setup time to ***x*clk**— point-to-point[1], [3]	10, 12	–	ns
T_h	Input signal hold time from ***x*clk**[1]	0	–	ns

[1]All primary interface signals are used by **p_clk** and all secondary interface signals are used by **s_clk**.

[3]Point-to-point signals are **p_req_l**, **s_req_l<7:0>**, **p_gnt_l**, **s_gnt_l<7:0>**, **s_dispst_l**, **s_bufne_l**, and **s_cfn_l**.
All other PCI signals are shared.
All ***x*gnt_l** signals, **s_dispst_l**, and **s_cfn_l** have a setup time of 10 ns.
***x*req_l** has a setup time of 12 ns.

Figure 7–3 AC Timing Waveforms

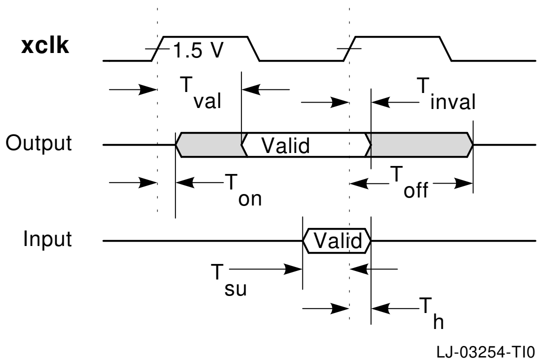


Table 7–6 shows the timing specifications for ***x*rst_l**.

Table 7–6 *x*rst_l Timing Specifications

Symbol	Parameter	Minimum	Maximum	Units
T_{prst}	**p_rst_l** active time after power stable	1	–	µS
$T_{prst-clk}$	**p_rst_l** active time after **p_clk** stable	100	–	µS
$T_{prst-off}$	**p_rst_l** active to output float delay	–	40	ns
T_{srst}	**s_rst_l** active after **p_rst_l** assertion	–	40	ns
$T_{srst-on}$	**s_rst_l** active time after **s_clk** stable	100	–	µS
$T_{srst-off}$	**s_rst_l** active to secondary output float delay	–	40	ns

A

Technical Support, Ordering, and Associated Literature

This appendix describes how to obtain DECchip information and technical support, as well as how to order DECchip products and associated literature.

A.1 Calling the DECchip Information Line for Information and Technical Support

Use the following numbers to call the DECchip Information Line:

United States and Canada	**1–800–332–2717**
TTY (United States only)	**1–800–332–2515**
Outside North America	**+1–508–568–6868**

A.2 Ordering DECchip Products

To order the DECchip 21050 PCI-to-PCI Bridge, contact your local Digital sales office. Your sales representative may be able to help you take advantage of discounts and volume pricing.

You can order the following DECchip products from Digital.

Product	Order Number
DECchip 21050 PCI-to-PCI Bridge	21050–AA

A.3 Ordering Associated DECchip Literature

The following table lists some of the DECchip literature that is available. For a complete list and for information about ordering, contact the DECchip Information Line.

Title	Order Number
DECchip 21050 PCI-PCI Bridge Product Brief	EC–N0569–72
DECchip 21050 PCI-PCI Bridge Configuration: An Application Note	EC–Q9Y4A–TE
DECchip 21050 PCI-PCI Bridge Hardware Implementation: An Application Note	EC–Q9Y3A–TE

A.4 Ordering Third-Party Literature

You can order the following third-party literature directly from the vendor.

Title	Vendor
PCI Local Bus Specification, Revision 2.0	PCI Special Interest Group M/S HF3–15A 5200 N.E. Elam Young Pkwy Hillsboro, Oregon 97124–6497 (503) 696–2000